SINERSYS

Informática Integral Empresarial, S.A. de C.V.

C0202NA1

CURSO DE PROGRAMACION EN NATURAL

Marzo, 1997

CONTENIDO

I. OBJETIVO

II. ANTECEDENTES

III. INTRODUCCION

IV. PROGRAMACION NATURAL.

1. Programación básica.
   - 1.1. Introducción al ambiente Natural.
   - 1.2. Estructura del programa .
   - 1.3. Tipos de datos y objetos.
   - 1.4. Instrucciones básicas.
   - 1.5. Accesos simples a la base de datos
   - 1.6. Variables del sistema.

2. Creación de reportes y cortes.
   - 2.1. Write.
   - 2.2. Write title. / Write trailer
   - 2.3. Display.
   - 2.4. Mascaras de edición.
   - 2.5. Format
   - 2.6. Cortes de control.

3. Editores.
   - 3.1. Editor de programas.
   - 3.2. Editor de áreas de trabajo.
   - 3.3. Editor de pantallas.

4. Implementación de pantallas.
   - 4.1. Invocando una pantalla.
   - 4.2. Uso de teclas de función (PF's).
   - 4.3. Técnicas de validación.
   - 4.4. Técnicas de paginación.

5. Modularidad.
   - 5.1 Subprogramas.
   - 5.2 Subrutinas.
   - 5.3 Mapas.
   - 5.4 Area local de datos.
   - 5.5 Area global de datos.
   - 5.6 Llamadas a otros programas.
   - 5.7 Helproutine.

6. Vistas, accesos y manipulación a la base de datos.
   - 6.1. Vistas de usuario.
   - 6.2. Accesos complejos (escape,etiquetas,sort,accept,..).
   - 6.3. Instrucciones de actualización.
   - 6.4. Accesos a archivos secuenciales.

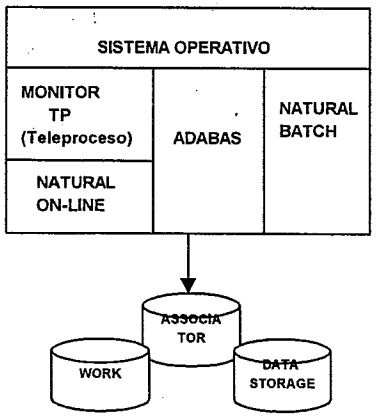
7. Manipulación de datos.
   - 7.1. Definición y manejo de arreglos.
   - 7.2 Uso de la variable C*.
   - 7.3. Uso de la instrucción MOVE.
   - 7.4. Uso de las instrucciones compress, separate, examine.
   - 7.5. Funciones de Natural.
   - 7.6 Redefinición de variables

V. EJEMPLOS.

1. PROGRAMACION BASICA.

1.1 Introducción al ambiente Natural.

Natural es un lenguaje 4GL (4th. Generation Language). Generalmente trabaja con la base de datos ADABAS (Adaptable DAta BAse System) como se puede apreciar en el siguiente diagrama.



En Natural existen dos modos de codificación:

- REPORTING (Modo reporte)
- STRUCTURED (Modo estructurado)

Cuyas diferencias Básicas son:

MODO	DEFINE DATA	SINTAXIS (ejemplos)	
REPORTING Inicios del Natural	Es opcional	FIND ... ... LOOP	IF DO ... ... DO-END ELSE ...
STRUCTURED Versiones Posteriores	Es necesaria	FIND ... ... END-FIND	IF ... ... ELSE ... ... END-IF

En este manual se utilizará una nomenclatura especial al dar la sintaxis de las diversas instrucciones que se utilizan en Natural como se lista en la siguiente tabla:

[ ]	Corchetes	Cláusulas opcionales
{ }	Llaves	Elemento requerido
...	Puntos suspensivos	Sucesivamente
abc	Letras minúsculas	Información variable dada por el usuario
ABC	Letras mayúsculas	Palabra reservada de Natural o palabra llave
( )	Paréntesis	Número de reporte, número de líneas
'	Apóstrofe	Representa texto
/	Diagonal	Salto de líneas

1.2. Estructura del programa.

La estructura básica de un programa está integrada por dos secciones: definición de variables y codificación del programa. La primera inicia con la sentencia DEFINE DATA y a continuación se declaran las variables globales, las vistas de los archivos de la base de datos, archivos secuenciales de entrada y/o salida, parámetros y las variables de trabajo a utilizar en el programa; cerrándose esta sección con el END-DEFINE. La segunda sección, donde se escriben las sentencias Natural, debe terminar obligatoriamente con la sentencia END, como se muestra a continuación:

```
DEFINE DATA
    Definición de variables
....
.....
END-DEFINE
Sentencias Natural
.....
.....
END
```

INTRODUCCION.

El lenguaje de programación NATURAL, es de formato libre; es decir, el programador decide libremente los nombres de variables, archivos, secciones y todos los elementos que conforman el programa. Esto trae consigo que cada persona adquiera un estilo propio de programación, de manera similar a como cada quien tiene un estilo propio de hablar.

A consecuencia de esto, en un proyecto donde participen varios programadores, se tendría un concierto de estilos diferentes de programación y como resultado, cada quien solo podría dar mantenimiento a los programas elaborados por el mismo.

El problema es cada vez mayor, considerando que una persona no estará siempre en el mismo lugar, así que en un proyecto donde desfilen unas 50 personas, llegaría en momento en que el sistema serie inentendible.

La solución a este problema, es que todas las personas involucradas en el mismo proyecto, se apeguen a un solo estilo de programación independientemente de sus estilos personales. Es importante señalar que dicho estilo o infraestructura, debe ser lo más flexible posible, para permitir la codificación de cualquier problema de programación que se tenga.

Además, la infraestructura de programación debe contemplar la estandarización no solo del estilo de programación, sino también de los algoritmos y situaciones más comúnmente presentadas en el ambiente de programación.

1.3. Tipos de Datos y objetos

Los tipos de datos que soporta Natural se listan a continuación:

Formato	Longitud (en bytes)	Longitud por default
A (Alfanumérico)	1 - 253	10
B (Binario)	1 - 126	-
C (Atributo de control)	(2)	-
D (Date)	(4) (P6)	-
F (Flotante)	4,8	-
I (Entero)	1,2,4	-
L (Lógico)	(1)	-
N (Numérico)	1 - 27	7
P (Numérico Empacado)	1 - 27	7
T (Time)	(7) (P12)	-

Los tipos de objetos usados en Natural son :

- Programas
- Subprogramas
- Subrutinas
- Rutinas de Ayuda (Helproutines)
- Mapas
- Areas Locales de datos
- Areas Globales de datos
- Areas de Parámetros
- Copycodes (partes pequeñas de código fuente).

En capítulos posteriores se examinarán a detalle cada uno de estos objetos.

1.4. Instrucciones básicas.

1.4.1 Estructuras de Control

En esta sección codificaremos algunos programas simples en Natural, que nos darán una introducción a la sintaxis del lenguaje mediante el uso de estructuras de control.

La ejecución condicionada se realiza cuando un proceso depende del resultado de una condición.

```
IF condición THEN
   proceso A

ELSE
   proceso B

END-IF
```

Ejemplo :

```
0010 DEFINE DATA LOCAL
0020 01 ÑWN-NUMERO          (N02)
0030 END-DEFINE
0040 *******************************************
0050 ** INICIO DE PROGRAMA                     **
0060 *******************************************
0070 INPUT ÑWN-NUMERO
0080 IF ÑWN-NUMERO > 10 THEN
0090    WRITE ' NUMERO MAYOR A 10 '
0100 ELSE
0110    WRITE ' NUMERO MENOR O IGUAL A 10 '
0120 END-IF
0130 *******************************************
0140 ** FIN DE PROGRAMA                        **
0150 *******************************************
0160 END
```

```
0010 DEFINE DATA LOCAL
0020 01 ÑWN-NUMERO          (N02)
0030 END-DEFINE
0040 *******************************************
0050 ** INICIO DE PROGRAMA                     **
0060 *******************************************
0070 INPUT ÑWN-NUMERO
0080 IF ÑWN-NUMERO <=20 THEN
0090    IF ÑWN-NUMERO >=10 THEN
0100 WRITE 'EL NUMERO ESTA EN EL RANGO DE 10 A 20'
0110    ELSE
0120       WRITE 'EL NUMERO ES MENOR A 10'
0130    END-IF
0140 END-IF
0150 *******************************************
0160 ** FIN DE PROGRAMA                        **
0170 *******************************************
0180 END
```

La ejecución condicionada múltiple se presenta cuando un proceso depende del resultado de la evaluación de más de dos condiciones. La codificación de esta instrucción puede ser de dos formas, la primera en que la condición de evaluación es la misma para todas las opciones, y la segunda en la cual cambia la condición de evaluación para cada opción.

Primera opción

```
DECIDE ON FIRST VALUE OF  condición
   VALUE  1
        proceso 1
   VALUE  2
        proceso 2
   VALUE  3
        proceso 3
   NONE
        proceso default
END-DECIDE
```

Ejemplo :

```
0010 DEFINE DATA
0020 LOCAL
0030 01 ÑWN-I        (N02)
0040 END-DEFINE
0050 ******************************************
0060 ** INICIO DE PROGRAMA                    **
0070 ******************************************
0080 INPUT ÑWN-I
0090 DECIDE ON FIRST VALUE OF ÑWN-I
0100    VALUE 1 : 10
0110       WRITE 'VALOR ENTRE 1 Y 10'
0120    VALUE 11 : 20
0130       WRITE 'VALOR ENTRE 11 Y 20'
0140    VALUE 21 : 90
0150       WRITE 'VALOR ENTRE 21 Y 90'
0160    NONE
0170       WRITE 'VALOR ENTRE 91 Y 99'
0180 END-DECIDE
0190 ******************************************
0200 ** FIN DE PROGRAMA                       **
0210 ******************************************
0220 END
```

Segunda opción

```
DECIDE FOR FIRST CONDITION
   WHEN  condición 1
        proceso 1
   WHEN  condición 2
        proceso 2
   WHEN  condición 3
        proceso 3
   NONE
        proceso default
END-DECIDE
```

Ejemplo :

```
0010 DEFINE DATA
0020 LOCAL
0030 01 ÑWN-I        (N02)
0040 01 ÑWA-OPCION   (A01)
0050 END-DEFINE
0060 ******************************************
0070 ** INICIO DE PROGRAMA                    **
0080 ******************************************
0090 INPUT ÑWN-I ÑWA-OPCION
0100 DECIDE FOR FIRST CONDITION
0110    WHEN ÑWA-OPCION = 'A' AND ÑWN-I <= 50
0120       WRITE 'OPCION A Y VALOR ENTRE 1 Y 50'
0130    WHEN ÑWA-OPCION = 'A' AND ÑWN-I > 50
0140       WRITE 'OPCION A Y VALOR ENTRE 51 Y 99'
0150    WHEN ÑWA-OPCION = 'B'
0160       WRITE 'OPCION B'
0170    NONE
0180       WRITE 'OPCION INCORRECTA'
0190 END-DECIDE
0200 ******************************************
0210 ** FIN DE PROGRAMA                       **
0220 ******************************************
0230 END
```

La ejecución cíclica ocurre cuando un proceso se repite varias veces. Existen tres tipos de ejecución cíclica: la estructura "ejecuta-mientras", la estructura "repite-hasta" y la estructura "desde-hasta" .

La estructura "ejecuta-mientras" consiste en evaluar la condición antes de realizar el proceso y este continuará ejecutándose hasta que la condición no se cumpla. Es importante notar que el proceso no se ejecuta si la condición de evaluación se cumple al inicio.

```
REPEAT WHILE condición
   proceso

END-REPEAT
```

La estructura "repite-hasta" consiste en realizar el proceso y después evaluar la condición, el proceso continuará ejecutándose hasta que la condición

se cumpla. Es importante notar que en esta estructura el proceso se efectúa al menos una vez y después se evalúa la condición.

```
REPEAT
   proceso

   UNTIL condición
END-REPEAT
```

La estructura "desde-hasta" nos permite realizar un ciclo con un número predeterminado de iteraciones. Esta estructura nos sirve cuando sabemos con anterioridad el número de ciclos que deseamos que un proceso realice.

```
FOR operando1 = operando2 TO operando3
   proceso
END-FOR
```

```
0010 DEFINE DATA
0020 LOCAL
0030 01 ÑWN-I        (N02)  INIT <0>
0040 01 ÑWN-J        (N02)  INIT <0>
0050 END-DEFINE
0060 ******************************************
0070 ** INICIO DE PROGRAMA                    **
0080 ******************************************
0090 REPEAT WHILE ÑWN-I <= 10
0100    ADD 1         TO ÑWN-I
0110    WRITE 'CONTADOR =' ÑWN-I
0120 END-REPEAT
0130 ******************************************
0140 REPEAT
0150    ADD 1         TO ÑWN-J
0160    WRITE 'CONTADOR =' ÑWN-J
0170    UNTIL ÑWN-J >= 10
0180 END-REPEAT
0190 ******************************************
0200 FOR ÑWN-I = 1 TO 10
0210    WRITE 'CONTADOR =' ÑWN-I
0220 END-FOR
0230 ******************************************
0240 ** FIN DE PROGRAMA                       **
0250 ******************************************
0260 END
```

1.4.2 Instrucciones para operaciones aritméticas

Para poder realizar operaciones aritméticas, Natural cuenta con las siguientes instrucciones:

1. Suma

ADD [ROUNDED] {operando1} TO / GIVING operando2

Operandos:

Cada operando deberá contener un valor correcto cuando se ejecute el ADD. Para un procesamiento más óptimo, las variables de una suma podrían ser definidas con formato P y de la misma precisión.

Campo de Resultado(operando2):

Si la palabra llave TO es utilizada, operando2 será incluido en la operación de adición y contendrá el resultado de la misma. Si la palabra llave GIVING es usada, operando2 únicamente almacenará el resultado.

Opción ROUNDED:

Causa que el resultado sea redondeado al entero más cercano (cuando se trabaja con decimales).

2. Resta

SUBTRACT [ROUNDED] {operando1}...FROM operando2 [GIVING operando3]

Opción ROUNDED:

Causa que el resultado sea redondeado al entero más cercano (cuando se trabaja con decimales).

Operandos:

Cada operando deberá contener un valor permitido cuando se ejecute el SUBTRACT. Para un procesamiento más óptimo, las variables en una operación de resta podrían ser definidas con formato P y de la misma precisión.

GIVING:

Si es utilizada esta cláusula, Operando2 no será modificado y el resultado será almacenado en operando3. Si se omite la cláusula GIVING, el resultado será almacenado en Operando2.

3. Multiplicación

MULTIPLY [ROUNDED] operando1 BY operando2 [GIVING operando3]

Opción ROUNDED:

Causa que el resultado sea redondeado al entero más cercano (cuando se trabaja con decimales).

Campo de resultado:

El resultado de la multiplicación podrá ser almacenado en el operando1 o en el operando3. Si la cláusula GIVING no es usada, el resultado es almacenado en operando1; de otra forma, operando1 no será modificado y el resultado será almacenado en operando3. Si operando1 es una constante, es necesaria la cláusula GIVING.

4. División

DIVIDE [ROUNDED] operando1 INTO operando2 [GIVING operando3] [REMAINDER operando3]

Campo de resultado:

Este podrá ser una variable definida por el usuario o un campo de una base de datos. Si una constante numérica es utilizada en el operando2, la cláusula GIVING es requerida.

ROUNDED:

Redondea el resultado.

GIVING:

Si es usada, operando2 no será modificado y el resultado será almacenado en operando3; en otro caso, el resultado será almacenado en operando2.

División entre Cero:

Cuando se realiza, un mensaje de error o un resultado igual a cero será regresado según el valor que tenga el parámetro ZD de la sesión de Natural.

Opción REMAINDER:

Deja el residuo de la división en un campo previamente especificado.

5. Operaciones múltiples

COMPUTE [ROUNDED]{operando1} = ... expresión aritmética

Función:

Es utilizado para realizar operaciones aritméticas.

Campo de Resultado(operando1):

El campo especificado a la izquierda del signo igual, contendrá el resultado de la operación aritmética.

Expresiones Aritméticas:

Consisten de una o más constantes, campos de bases de datos y/o variables definidas por el usuario. Las funciones aritméticas de Natural podrán también ser utilizadas como operandos aritméticos. Los operandos deberán ser definidos con alguno de los siguientes formatos: N, P, I, F, D, T. Para optimizar el proceso, es recomendable definir las variables de usuario con el formato P y con la misma precisión.

Los siguientes operadores son válidos dentro de la expresión aritmética:

( ) Paréntesis
** Exponente
* Multiplicación
/ División
+ Adición o suma
- Sustracción o resta.

Cada operador deberá ser precedido y seguido por al menos un blanco.

Orden de Precedencia:

Paréntesis
Exponente
Multiplicación / división (de izquierda a derecha)
Suma / Resta (de izquierda a derecha)

1.5. Accesos simples a la base de datos.

Existen tres tipos de accesos a la base de datos, los accesos secuenciales, los accesos aleatorios, y el acceso directo. Los dos primeros crean un proceso iterativo, la misma instrucción controla el ciclo, y solo el acceso directo no crea un ciclo.

1.5.1 Accesos Secuenciales.

Para este tipo de accesos se pueden obtener los datos con los siguientes métodos:

- Secuencia Física. (Read physical)
- Secuencia por ISN. (Read by ISN)
- Secuencia Lógica. (Read by descriptor)
- HISTOGRAM.

Estos métodos varían en tres puntos principalmente:

- Las estructuras que se usan de la base de datos.
- La secuencia en la que los datos son obtenidos.
- Se da un valor de inicio o no.

1.5.1.1 Read Physical.

- Utiliza la estructura de almacenamiento de datos.
- La secuencia en la que los datos son obtenidos es como están grabados en la base de datos
- No se puede dar un valor de inicio.

```
READ vista de archivo PHYSICAL
    proceso
  ...
END-READ
```

1.5.1.2 Read By ISN.

- Utiliza las estructuras de convertidor de direcciones y almacenamiento de datos.
- La secuencia en la que los datos son obtenidos es en orden ascendente por ISN.
- Si se puede dar un valor de inicio.

```
READ vista de archivo BY ISN
    proceso
  ...
END-READ
```

1.5.1.3 Read By Descriptor.

- Utiliza las estructuras de lista invertida, convertidor de direcciones y almacenamiento de datos.
- La secuencia en la que los datos son obtenidos es en orden ascendente por el valor inicial del descriptor.
- Si se puede dar un valor de inicio.

```
READ vista de archivo BY campo-descriptor
    proceso
  ...
END-READ
```

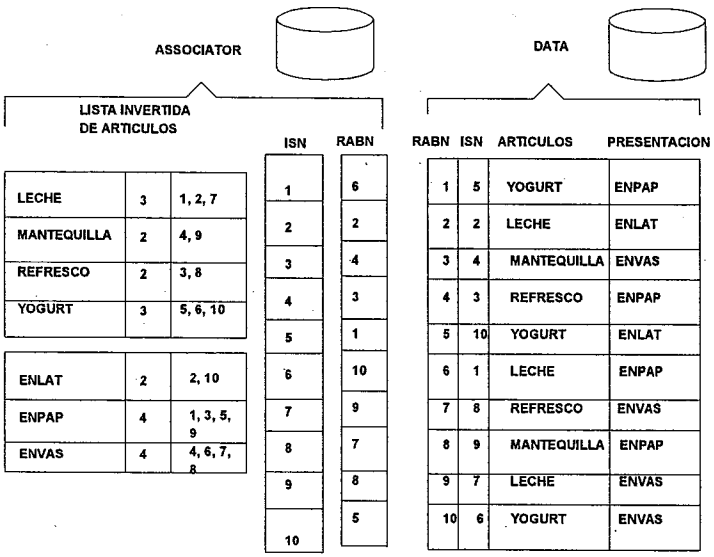
1.5.1.4 Histogram.

- Utiliza la estructura de lista invertida.
- Solo se obtiene el valor del descriptor.
- La secuencia en la que se obtienen los datos es en orden ascendente por el valor inicial del descriptor.
- Si se puede dar un valor de inicio.

```
HISTOGRAM vista de archivo FOR campo-descriptor
    proceso
  ...
END-HISTOGRAM
```

Ejemplos :

Dada la siguiente estructura del archivo de artículos :



EJEMPLOS: (Utilizando la estructura 1)

Para los casos de READ:

Crear una lista de ISN's en el orden en el que los datos serán obtenidos.

READ PHYSICAL

```
0010 DEFINE DATA
0020 LOCAL
0030 01 VIEW-ARTICULO VIEW OF ART-COM
0040    02 NOMBRE-ART
0050    02 CVE-PRESNT
0060 END-DEFINE
0070 ******************************************
0080 ** INICIO DE PROGRAMA                    **
0090 ******************************************
0100 READ VIEW-ARTICULO PHYSICAL
0110    WRITE NOMBRE-ART CVE-PRESNT *ISN
0120 END-READ
0130 ******************************************
0140 ** FIN DE PROGRAMA                       **
0150 ******************************************
0160 END
```

ARTICULO	CLAVE DE PRESENTACION	ISN
YOGURT	ENPAP	5
LECHE	ENLAT	2
MANTEQUILLA	ENVAS	4
REFRESCO	ENPAP	3
YOGURT	ENLAT	10
LECHE	ENPAP	1
REFRESCO	ENVAS	8
MANTEQUILLA	ENPAP	9
LECHE	ENVAS	7
YOGURT	ENVAS	6

READ BY ISN

```
0100 READ VIEW-ARTICULO BY ISN
0110    WRITE *ISN
0120 END-READ
```

1	2	3	4	5	6	7	8	9	10

READ BY descriptor

Por Artículo

```
0100 READ VIEW-ARTICULO BY NOMBRE-ART
0110    WRITE *ISN
0120 END-READ
```

1	2	7	4	9	3	8	5	6	10

Por Clave de presentación

```
0100 READ VIEW-ARTICULO BY CVE-PRESNT
0110    WRITE *ISN
0120 END-READ
```

2	10	1	3	5	9	4	6	7	8

HISTOGRAM by artículo

```
0100 HISTOGRAM VIEW-ARTICULO BY NOMBRE-ART
0110    WRITE NOMBRE-ART *NUMBER
0120 END-HISTOGRAM
```

Nombre-art	Contador
Leche	3
Mantequilla	2
Refresco	2
Yoghurt	3

1.5.2 Accesos Aleatorios.

Para este tipo de accesos se utiliza el FIND y las estructuras que utiliza son: la lista invertida, el convertidor de direcciones y el almacenamiento de datos; la forma en que se obtienen los datos es en forma ascendente por el valor del ISN.

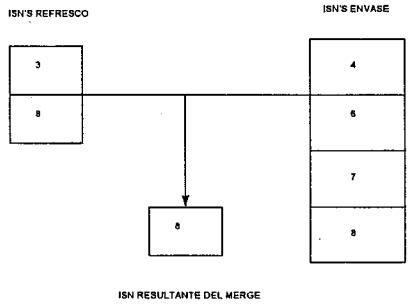
Si se puede dar un valor inicial.

Los operadores que se pueden utilizar para los criterios de selección son:

EQ	o	=	IGUAL
NE	o	¬=	DIFERENTE
GT	o	>	MAYOR QUE
LT	o	<	MENOR QUE
GE	o	>=	MAYOR O IGUAL A
LE	o	<=	MENOR O IGUAL A

EJEMPLO: (Utilizando la estructura 1)

- Obtener la lista de ISN's donde el Producto = 'REFRESCO' y Clave presentación = 'ENVAS'



CUADRO COMPARATIVO

METODO DE ACCESO	ESTRUCTURAS USADAS	SECUENCIA OBTENIDA	VALOR INICIAL
READ PHYSICAL	Almacenamiento de datos.	Secuencia Física.	No
READ BY ISN	Convertidor de direcciones. Almacenamiento de datos.	Secuencia de ISN's.	Opcional
READ LOGICAL	Listas invertidas. Convertidor de direcciones. Almacenamiento de datos.	Secuencia del valor del descriptor.	Opcional
HISTOGRAM	Listas invertidas.	Secuencia del valor del descriptor.	Opcional
FIND	Listas invertidas. Convertidor de direcciones. Almacenamiento de datos.	Secuencia de ISN's	Opcional

1.6 Variables del sistema

A continuación se listan las variables de sistema de Natural más utilizadas en los programas:

Variable de sistema	Formato/ Longitud	Explicación
*COUNTER [(r)]	P8	El número de veces que un ciclo procesado iniciado por una sentencia FIND, READ o HISTOGRAM ha sido ejecutado. (r) es usado para indicar la etiqueta o número de sentencia del FIND, READ o HISTOGRAM. Si (r) no es especificada, *COUNTER representa el número de veces que ha sido ejecutado el ciclo de proceso activo actualmente. *COUNTER no se incrementa si un registro ha sido rechazado como resultado de un criterio especificado en la cláusula WHERE.
*DATA	N3	El número de elementos de datos en el Stack (pila) de Natural, los cuales están disponibles para la próxima sentencia INPUT como datos de entrada. *DATA contendrá cero cuando el stack este vacío. Un valor de -1 indica que el siguiente elemento en el stack es un comando o el nombre de una transacción de Natural.
*DEVICE	A8	El tipo/modo de dispositivo desde el cual fue invocado Natural BATCH Modo batch COLOR 3279 Compatibilidad 3278 (Dispositivo con soporte de atributo extendido. VIDEO 3270 Dispositivos de pantalla TTY Teletipo u otro dispositivo de start/stop PC Dispositivo de PC.
*ERROR-LINE	N4	El número de línea de la sentencia en el que ha ocurrido un error.
*ISN [(r)]	P8	Obtiene el número secuencial interno ADABAS del registro que actualmente se esta procesando dentro de un ciclo iniciado por FIND o un READ. (r) es usado para indicar la etiqueta o el número de la sentencia en donde está el FIND o el READ. Si (r) no es especificada *ISN representa el ISN del registro procesado por el ciclo activo.
*LIBRARY-ID	A8	Nombre de la librería donde actualmente se está trabajando.
Variable de sistema	Formato/ Longitud	Explicación
*LINE-COUNT [(rep)]	P5	El número de líneas dentro de la página actual. Si se producen múltiples reportes en el programa, (rep) es utilizado para especificar la identificación del reporte para el cual el número de línea actual es requerido. El valor es incrementado siempre en 1 para la siguiente línea a ser impresa. El valor de *LINE-COUNT es actualizado durante la ejecución de un WRITE, SKIP o DISPLAY. El NEWPAGE y el EJECT causan que *LINE-COUNT sea puesto en cero.
*NUMBER [(r)]	P8	El número de registros seleccionados como resultado de un FIND (resultantes de un cláusula WITH), o el número de valores seleccionados como resultado de una sentencia HISTOGRAM. (r) Es usado para indicar la etiqueta o el número de la sentencia en la cual se encuentra el FIND o el HISTOGRAM. Si no se especifica, *NUMBER representa el número de registros seleccionados por un FIND o un HISTOGRAM usado para iniciar ciclo activo actualmente.
*PAGE-NUMBER [(rep)]	P5	El valor actual para el número de página de un reporte. Si múltiples reportes son producidos por el programa, (rep) es usado para especificar la identificación del reporte para el cual el número de página actual es requerido. Esta variable es definida por Natural en el momento que se inicia el formato para el reporte, su valor siempre es incrementado en 1 para la siguiente página iniciada por un WRITE, DISPLAY, SKIP o NEWPAGE. El EJECT no incrementa el valor del *PAGE-NUMBER.
*PF-KEY	A4	Tecla que fue presionada y fue la última entrada recibida por Natural. PA1 - 3 Teclas de atención de programa 1 - 3 PF1 - 24 Teclas de función de programa 1 - 24 ENTR Tecla ENTER CLR Tecla CLEAR
*PROGRAM	A8	El nombre del programa Natural en ejecución actual.
*USER	A8	El identificador del usuario tomado del Security Logon de Natural.
*USER-NAME	A32	Si el sistema de seguridad de Natural se encuentra instalado, esta variable contendrá el nombre del usuario que se encuentra firmado actualmente en Natural. El valor por default es 'SYSTEM'.
*DATD	A8	Fecha actual en formato DD.MM.YY
*DATE	A8	Fecha actual en formato DD/MM/YY
*DATG	A15	Fecha actual en formato DDnombre-de-mesYYYY (Fecha gregoriana)
*DATI	A8	Fecha actual en formato YY-MM-DD
*DATJ	A5	Fecha juliana actual en formato YYDDD
*DATN	N8	Fecha actual en formato YYYYMMDD
*DATU	A8	Fecha actual en formato MM/DD/YY
*DATX	D	Fecha actual en formato interno de fecha
*TIME	A10	Hora del día en formato HH:MM:SS.T
*TIMESTMP	B8	Valor del reloj interno de la máquina
*TIMN	N7	Hora del día en formato HHMMSST
*TIMX	T	Hora del día en formato interno de tiempo

2. CREACION DE REPORTES Y CORTES

Para la creación de reportes, Natural utiliza esencialmente 2 sentencias para imprimir los datos con el formato requerido:

- WRITE
- DISPLAY

Cada una de estas sentencias tiene sus propias características y parámetros que suministran las herramientas necesarias para obtener el formato del reporte deseado; por lo que dependiendo de los requerimientos solicitados, se debe utilizar una u otra sentencia con sus respectivos parámetros. A continuación se explicará brevemente la sintaxis y uso de las sentencias WRITE y DISPLAY.

2.1 WRITE

La sentencia WRITE se utiliza para producir la salida de datos en un formato libre en pantalla o en impresión. Tiene las siguientes características:

- Permite el desbordamiento de línea, lo cual significa que si se excede el tamaño de línea, la información continúa en la siguiente línea.
- No escribe encabezados de columna automáticamente.
- Los valores u ocurrencias de un arreglo se presentan en forma horizontal.
- Como resultado de una sentencia WRITE, Natural genera una línea de títulos sencilla que contiene el número de página, la hora y la fecha.

Sintaxis:

WRITE [(rep)] [NOTITLE] [(parámetros)] operando1

Especificación de Reporte (rep)

WRITE (rep) especifica el número del reporte, en caso de que el programa emita varios reportes. Su valor debe estar comprendido entre 1 y 31. También puede ser un nombre lógico asignado previamente mediante la sentencia DEFINE PRINTER. Si no se especifica, el valor será el del primer reporte (0).

NOTITLE

Esta sentencia suprime la línea de título que Natural genera por cada página que resulte de una sentencia WRITE.

Parámetros

Uno o más parámetros podrán especificarse inmediatamente después del WRITE o un elemento a ser desplegado. Estos parámetros reemplazarán a aquéllos que han sido especificados con la sentencia FORMAT. Si se especifica más de un parámetro, debe haber por lo menos un espacio en blanco entre cada uno de ellos.

NX	Esta notación sirve para insertar 'n' espacios entre columnas. 'n' no puede ser cero WRITE NAME 5X SALARY
nT	Con esta notación se empieza a imprimir el campo a partir de la posición 'n' (tabulación). No permite posicionarse antes de 'n' WRITE 25T NAME 50T SALARY (NAME empezará a ser impreso en la posición 25 y SALARY en la posición 50).
x/y	Posiciona el siguiente elemento en la línea x, columna y.
T* nombre del campo	Esta notación es utilizada para posicionar un lugar de impresión específico de un campo usado en una sentencia DISPLAY previa.
P* nombre del campo	Esta notación posiciona un lugar de impresión específico y línea de un campo definido en una sentencia DISPLAY previa. Se emplea frecuentemente con el modo de impresión vertical.

Después de seleccionar alguno de los parámetros anteriores, se pueden utilizar los siguientes:

'texto'	Si se escribe inmediatamente antes de un campo, 'texto' sustituirá el encabezado de la columna. WRITE 'EMPLEADO' NAME		
'c' (n)	Igual que 'texto' excepto que el carácter 'c' es puesto 'n' veces WRITE '*' (5) '=' NAME		
atributos	Estos se colocan después de 'texto' o 'c'. Indican los atributos de impresión y color a ser usado por el texto o campo a desplegar.		
	Valor	Significado	Explicación
	C	Cursiva/itálica	El valor del campo será desplegado cursivo/itálico
	U	Subrayado	El valor del campo será desplegado subrayado
	L	Justificado a la izquierda	El valor del campo a ser desplegado se justifica a la izquierda
	R	Justificado a la derecha	El valor del campo a ser desplegado se justifica a la derecha
	Z	Supresión de ceros	Los valores numéricos son desplegados con supresión de ceros, justificado a la derecha.
'-'	Cuando se escribe antes de un campo, '-' suprime la generación de un nombre de campo como texto tecleado.		
'='	Si se escribe antes de un campo, '=' resulta en el desplegado del encabezado de un campo seguido por su contenido.		
/	Cuando se escribe entre 2 campos o elementos de texto, '/' causa el posicionamiento al inicio de la siguiente línea de impresión. WRITE NAME / SALARY		

Operando1

Se refiere al nombre del campo a ser escrito. Los superdescriptores, subdescriptores, descriptores fonéticos e hiperdescriptores de ADABAS no podrán ser utilizados.

WRITE Usando formas predefinidas

Sintaxis:

WRITE [(rep)] [NOTITLE] [USING] FORM operando1
[{operando2}...]
MAP

FORM /MAP

Esta opción indica que se usará una forma de layout previamente definida a través el editor de mapas de Natural.

No crea automáticamente una nueva página por cada forma.

operando1 :

Es el nombre de la forma o mapa a desplegar.

operando2 :

Se refiere al nombre del campo a escribir.

Si operando1 es una constante y operando2 es omitido, los campos son tomados desde el mapa fuente en el momento de la compilación.

Ejemplo del uso de la sentencia WRITE

```
0010 DEFINE DATA LOCAL
0020 01 VIEW-ART VIEW OF ARTICULOS-COMESTIBLES
0030    02  CVE-ART
0040    02  NOMBRE-ART
0050    02  DESCRIP
0060    02  CANT
0070 END-DEFINE
0080 ******************************************
0090 ** INICIO DE PROGRAMA                    **
0100 ******************************************
0110 READ VIEW-ART BY NOMBRE-ART
0120         STARTING FROM 'YOGURT'
0130           ENDING  AT 'YOGURT'
0140          WRITE CVE-ART 13X DESCRIP
0150            / 'CANTIDAD EN EXISTENCIA: '  25T CANT
0160 /*  Deja 13 espacios de separación  entre los campos cve-
art y
0170 /*  descrip
0180 /*  Imprime el campo cant a partir de la posición 25
0190                    /
0200 END-READ
0210 ******************************************
0220 ** FIN DE PROGRAMA                       **
0230 ******************************************
0240 END
```

Cuya salida se muestra a continuación:

```
PAGE  1                          11:32:45
30/03/97

LACYOG1              YOGURT NATURAL
CANTIDAD EN EXISTENCIA:  200

LACYOG3              YOGURT SEMIDESCREMADO
CANTIDAD EN EXISTENCIA:  150

LACYOG7              YOGURT CON FRUTAS
CANTIDAD EN EXISTENCIA:  350

LACYOG80             YOGURT CON FRUTAS TROPICALES
CANTIDAD EN EXISTENCIA:  110
```

2.2 WRITE TITLE / WRITE TRAILER

WRITE TITLE

La función de la sentencia WRITE TITLE es sustituir los títulos de la página que tiene por default, por los títulos de página especificados por el usuario. Es ejecutado cuando alguna nueva página es iniciada.

Sintaxis:

WRITE [(rep)] TITLE [LEFT [JUSTIFIED]] [UNDERLINED] [(parámetros)] operando1 [(parámetros)] [SKIP [operando2[LINES]]].

Especificación de Reporte (rep)

El manejo de esta notación equivalente a la descrita en la instrucción WRITE.

Restricciones

Esta sentencia podrá ser escrita una vez por reporte y no puede ser especificada dentro de un bloque de condición especial.

JUSTIFIED / UNDERLINED (Justificación y Subrayado)

En el formato de salida estándar, el título y las líneas de encabezado son centradas y sin subrayar. LEFT JUSTIFIED y/o UNDERLINED podrán ser especificados para sustituir los valores que se tienen por default. Si se especifica UNDERLINED, el carácter de subrayado es impreso debajo del título y abarca la longitud del tamaño de la línea.

Parámetros

Los parámetros son idénticos a los descritos en la sentencia WRITE.

Operando1

Representa el campo o campos a ser desplegados dentro del título.

SKIP (operando2)

SKIP podrá ser utilizado para imprimir líneas en blanco inmediatamente después de la línea de título. El número de líneas (operando2), podrá ser especificado como una constante numérica o como el contenido de una variable numérica.

WRITE TRAILER

La función del WRITE TRAILER es imprimir el texto o el contenido de las variables en la parte inferior de la página (pie de página) .

Sintaxis:

WRITE [(rep)] TRAILER [LEFT [JUSTIFIED]] [UNDERLINED] [(parámetros)] operando1 [(parámetros)] [SKIP [operando2 [LINES]]]

Restricción

Esta sentencia podrá ser especificada únicamente una vez por reporte y no podrá ser escrita dentro de un bloque de condición especial.

Proceso

Esta sentencia es ejecutada cuando se detecta una condición de fin de página o de fin de información, o como resultado de una sentencia SKIP o NEWPAGE. Pero no se ejecuta como resultado de una sentencia EJECT. La condición de fin de página es checada únicamente después de ejecutar un DISPLAY/WRITE completo. Si una sentencia DISPLAY/WRITE produce múltiples líneas de salida, el desbordamiento de una página física podría ocurrir antes de que la condición de fin de página sea ejecutada.

Tamaño Lógico de la Página

El tamaño lógico deberá ser menor que el tamaño físico de la página para asegurar que la información del pie de página aparezca en la parte inferior de la misma.

Especificación de reporte (rep)

El uso de esta notación es equivalente al explicado en la sentencia WRITE.

JUSTIFIED / UNDERLINED (Justificación y Subrayado.)

Por default, las líneas del pie de página son centradas y no subrayadas. LEFT JUSTIFIED y UNDERLINED podrán ser especificados para sustituir esos valores que se tienen por default. Si se especifica UNDERLINED, el caracter de subrayado es impreso debajo del título y corresponderá a la longitud del tamaño de la línea. NATURAL centra o justifica a la izquierda el texto antes de aplicar cualquier factor de espaciamiento o tabulación.

Operando1

Representa el campo o campos a ser impresos como pie de página.

Parámetros

Los parámetros son idénticos a los usados con la sentencia WRITE.

SKIP (operando2)

SKIP podrá ser utilizado para saltar líneas inmediatamente después del pie de pagina. El número de líneas en blanco podrá ser especificado como una constante numérica o como el contenido de una variable numérica.

Ejemplo de un programa que utiliza WRITE TITLE / WRITE TRAILER

```
0010 DEFINE DATA LOCAL
0020 01  VIEW-ARTICULOS1 VIEW OF ARTICULOS-COMESTIBLES
0030    02  CVE-ART
0040    02  NOMBRE-ART
0050    02  DESCRIP
0060    02  CVE-PRESNT
0070 END-DEFINE
0080 ******************************************
0090 ** INICIO DE PROGRAMA                    **
0100 ******************************************
0110 FORMAT PS=18
0120 WRITE TITLE 'ESTE ES EL TITULO DE PAGINA'
0130 WRITE TRAILER / 'ESTE ES EL TRAILER DE PAGINA'
0140 AT TOP OF PAGE
0150      WRITE / 'COMIENZO DE PAGINA'
0160       SKIP 1
0170 END-TOPPAGE
0180 READ VIEW-ARTICULOS1 BY NOMBRE-ART
0190           DISPLAY NOMBRE-ART (IS=ON) CVE-ART DESCRIP CVE-
PRESNT
0200 END-READ
0210 ******************************************
0220 ** FIN DE PROGRAMA                       **
0230 ******************************************
0240 END
```

Cuya salida es la siguiente:

```
                    ESTE ES EL TITULO DE PAGINA

COMIENZO DE PAGINA

NOMBRE-ART CVE-ART DESCRIP                          CVE-PRESNT
---------- ------- ------------------------------   ----------
LECHE       LACLEC3 LECHE ULTRAPASTEURIZADA          ENPAP
            LACLEC5 LECHE EVAPORADA LIGHT            ENLAT
            LACLEC7 LECHE SEMIDESCREMADA             ENPAP
MANTEQUILL LACMAN0 MANTEQUILLA CON SAL               ENPAP
            LACMAN3 MARGARINA                        ENPAP
QUESO       LACQUE2 QUESO CHIHUAHUA                  ENPAP
            LACQUE4 QUESO CREMA PARA UNTAR           ENPAP
REFRESCO   BEBCOC2 COCA COLA DE 2 LITROS             ENVAS
            BEBJUM1 JUMEX FRUTAS TROPICALES          ENLAT
            BEBPAU0 PAU PAU 100 ML.                  ENVAS
            BEBPEPL PEPSI COLA DE LATA               ENLAT

                    ESTE ES EL TRAILER DE PAGINA
```

2.3 DISPLAY

Función

La sentencia DISPLAY es usada para especificar los campos que van impresos en un reporte con el formato de columna. Una columna y su encabezado correspondiente, son creados por cada campo.

Sintaxis:

DISPLAY [(rep)] [opciones] [formato de salida] operando1 [(parámetros)]

Especificación de Reporte (rep)

(rep) Se utiliza para especificar el número del reporte si se producirán varios reportes en el programa. Su valor debe estar comprendido entre 1 y 31 o bien ser un nombre lógico, el cual se asigna previamente mediante la sentencia DEFINE PRINTER. Si no se especifica, el valor será el del primer reporte (0).

Opciones:

[NOTITLE] [NOHDR] [[AND] [GIVE] [SYSTEM] FUNCTIONS] [(parámetros)]

NOTITLE

Como resultado de una sentencia DISPLAY, Natural genera una línea de títulos sencilla, que contiene el número de página, la hora y la fecha. Esta línea podrá ser anulada por medio de la sentencia WRITE TITLE o podrá ser suprimida especificando la cláusula NOTITLE en la sentencia DISPLAY.

NOHDR

Esta cláusula omite los encabezados de las columnas.

GIVE SYSTEM FUNCTIONS

Esta cláusula habilita las funciones de sistema de Natural AVER, COUNT, MAX, MIN, NAVER, NCOUNT, NMIN, SUM, TOTAL. Estas son evaluadas cuando se ejecuta una sentencia DISPLAY conteniendo la cláusula GIVE SYSTEM FUNCTIONS. Únicamente un DISPLAY por reporte podrá contener esta cláusula. Las funciones se ejecutarán sólo como resultado de una condición de fin de página. Todas las funciones excepto TOTAL son inicializadas a cero cuando una nueva página se inicia.

Formato de salida

Para dar el formato de salida se deberán especificar uno o más parámetros encerrados dentro de paréntesis. Estos parámetros son similares a los explicados en la sentencia WRITE.

NX	Esta notación sirve para insertar 'n' espacios entre columnas. 'n' no puede ser cero.
NT	Esta notación sirve para tabular las posiciones de impresión. No permite colocarse antes de la posición 'n'.
T* nombre del campo	Esta notación define un lugar de impresión específico para un campo especificado en una sentencia DISPLAY previa.
P* nombre del campo	Define un lugar de impresión específico y línea para un campo especificado en una sentencia DISPLAY previa. Se usa frecuentemente con el modo de impresión vertical.

Después de seleccionar alguno de los parámetros anteriores, se pueden escribir los siguientes:

'texto'	Si se escribe inmediatamente antes de un campo, 'texto' sustituirá el encabezado de la columna. El carácter '/' antes de un campo causa que el encabezado para el campo sea suprimido.
'c' (n)	Igual que 'texto' excepto que el carácter 'c' es puesto 'n' veces
atributos	Estos se colocan después de 'texto' o 'c'. Indican los atributos de impresión y color a ser usado por el texto o campo a desplegar.

Valor	Significado	Explicación
C	Cursiva/itálica	El valor del campo será desplegado cursivo/itálico
U	Subrayado	El valor del campo será desplegado subrayado
L	Justificado a la izquierda	El valor del campo a ser desplegado se justifica a la izquierda
R	Justificado a la derecha	El valor del campo a ser desplegado se justifica a la derecha
Z	Supresión de ceros	Los valores numéricos son desplegados con supresión de ceros, justificado a la derecha.

Línea de avance

El símbolo ' / ', significa salto de línea. Puede escribirse entre dos campos.

Operando1

Se refiere al nombre del campo a ser escrito. Los superdescriptores, subdescriptores, descriptores fonéticos e hiperdescriptores de ADABAS no podrán ser utilizados.

Parámetros

Uno o más parámetros podrán ser especificados inmediatamente después del operando1, como los explicados en la sentencia WRITE.

Programa de ejemplo con la sentencia DISPLAY

```
DEFINE DATA LOCAL
01  VIEW-ARTICULOS1 VIEW OF ARTICULOS-COMESTIBLES
   02  CVE-ART
   02  NOMBRE-ART
   02  DESCRIP
   02  CVE-PRESNT
END-DEFINE
*********************************************
** INICIO DE PROGRAMA                      **
*********************************************
READ VIEW-ARTICULOS1 BY NOMBRE-ART STARTING FROM 'LECHE'
                                  ENDING AT 'MANTEQUILLA'
DISPLAY 4X 'ARTICULO' NOMBRE-ART 4X 'CLAVE' CVE-ART
            4X 'DESCRIPCION' DESCRIP 4X 'PRESENTACION' CVE-
PRESENT
             SKIP 1
END-READ
*********************************************
** FIN DE PROGRAMA                         **
*********************************************
END
```

Cuya salida es la siguiente:

```
PAGE1                                12:34:45            07/11/96

 ARTICULO                CLAVE                          DESCRIPCION
PRESENTACION
 --------      -------   --------------------------   ------------
  LECHE        LACLEC3   LECHE ULTRAPASTEURIZADA            ENPAP

  LECHE        LACLEC5   LECHE EVAPORADA LIGHT              ENLAT

  LECHE        LACLEC7   LECHE SEMIDESCREMADA               ENPAP

  MANTEQUILLA          LACMAN0   MANTEQUILLA CON SAL        ENPAP

  MANTEQUILLA          LACMAN3   MARGARINA                   ENPAP
```

Comparación entre las sentencias DISPLAY y WRITE

	DISPLAY	WRITE
Forma en que presenta la información	En columnas	En línea
Impresión automática de título de página	Sí	Sí
Impresión automática de encabezados	Sí	No
Centrado de encabezados	Sí	No
Espacio entre campos	Controlado automáticamente por SF, podría ser anulado con nX y nT	Podría usarse nX o nT
'/' propicia un salto de línea	En la misma columna	En la primera columna
Permite desbordamiento de línea	No	Sí

2.4 Máscaras de edición

Las máscaras de edición tienen la función de dar un formato a los campos de salida. Pueden utilizarse con la sentencia WRITE o DISPLAY. La edición de máscara podrá ser anulada por la especificación de una nueva edición de máscara o por el parámetro EM=OFF.

Existe una notación abreviada para la edición de máscaras con grandes valores. A continuación se lista un ejemplo para un campo numérico, hexadecimal y alfanumérico:

```
EM=9(4).9(5)      es equivalente a EM=9999.99999
EM=H(10)          es equivalente a EM=HHHHHHHHHH
EM=X(6)           es equivalente a EM=XXXXXX
```

Esta notación podrá ser usada únicamente con los caracteres '9', 'H', 'X' y 'Z'.

Edición de máscaras para campos numéricos.

Es utilizado para campos con formato N, P, I y F; los cuales deberán contener al menos un '9' o 'Z'. Si existen más 9's o Z's que el número de posiciones contenidas en el valor del campo, el número de posiciones a imprimir en la edición de máscara, será ajustado al número de dígitos definidos por el valor del campo. Por el contrario, si existen menos 9's o Z's, los dígitos ubicados más a la izquierda del punto decimal serán truncados. A continuación se listan los caracteres que podrán ser utilizados con la edición de máscaras para campos numéricos:

Carácter	Uso
9	Posición a ser impresa
Z	Supresión de ceros a la izquierda del valor numérico. La letra podrá ser especificada repetidamente para representar la supresión de ceros flotantes. Una Z no podrá ser escrita a la derecha del punto decimal. Por default un cero será impreso a la izquierda del punto decimal, aún cuando únicamente se especifiquen Z's.
+	Un signo flotante será impreso precediendo el número. El signo puede ser generado como un '+' o un '-' dependiendo del valor del campo.
-	Un signo menos será impreso precediendo el número si el valor del campo es negativo.
S	Se imprimirá un signo a la izquierda de la columna. Un '+' será impreso para valores positivos y un '-' para valores negativos.
N	Un signo '-' será impreso a la izquierda de la columna si el valor del campo es negativo.
.	Un punto, si es usado como el primer carácter, representa una posición decimal y es significativa.
H	Designa el inicio de una edición de máscara hexadecimal. Si una H aparece en la primera posición todos los otros caracteres que no sean H serán considerados caracteres de inserción.

Edición de máscaras para campos alfanuméricos

La edición de máscaras alfanuméricas es sólo para campos con formato A y deberá contener al menos una 'X', que representa un carácter a ser impreso. Todos los caracteres excepto los paréntesis son permitidos incluyendo caracteres de inserción. El número de posiciones a ser desplegado será ajustado a la longitud de la edición de máscara si ésta es menor que la del campo.

Edición de máscaras con formatos D (Date) y T (Time)

Los siguientes caracteres de edición de máscara se aplican a campos cuya definición sea el formato:

encuentren en las sentencias PRINT, WRITE, WRITE TITLE, WRITE TRAILER o DISPLAY.

Date:

Carácter	Uso
DD	Día
ZD	Día con supresión de cero
MM	Mes
ZM	Mes con supresión de cero
YYYY	Año con 4 dígitos
YY	Año con 2 dígitos
Y	Año con un dígito
WW	Número de la semana
ZW	Número de la semana con supresión de cero
JJJ	Día juliano
ZZJ	Día juliano con supresión de cero
N...N	Nombre del día (depende del idioma)
N(n)	Nombre del día usando una longitud 'n'
L...L	Nombre del mes (depende del idioma)
L(n)	Nombre del mes usando una longitud 'n'
R	Año en números romanos (máximo 13 dígitos)

Los siguientes caracteres de edición de máscara son para los campos cuya definición sea el formato Time:

Carácter	Uso
T	Décimas de segundo
SS	Segundos
ZS	Segundos con supresión de cero
II	Minutos
ZI	Minutos con supresión de cero
HH	Horas
ZH	Horas con supresión de cero
AP	Elemento AM/PM

2.5 FORMAT

Esta sentencia define los parámetros de salida y anula los valores que se tienen por default contenidos en el comando GLOBALS o que fueron definidos por el administrador de Natural. Pero a su vez, los parámetros definidos en la sentencia FORMAT podrán ser anulados por aquéllos que se

Los parámetros especificados con FORMAT permanecen activos hasta el final del programa o hasta que otra sentencia FORMAT es encontrada.

Dentro de un programa están permitidos múltiples FORMAT.

Sintaxis:

FORMAT [(rep)] {parámetros}...

Especificación de Reporte (rep)

(rep) especifica el número del reporte si se imprimirán diversos reportes en el programa. Su valor debe estar comprendido entre 1 y 31, o bien ser un nombre lógico el cual se asigna previamente con la sentencia DEFINE PRINTER. Si no se especifica, el valor será el del primer reporte (0).

Parámetros

Podrán ser especificados en cualquier orden y deberán estar separados por uno o más espacios. No podrá dividirse un parámetro en 2 renglones. Los principales parámetros que pueden utilizarse con la sentencia FORMAT y relacionados con reportes son los siguientes:

AL (Longitud Alfanumérica)

Se utiliza para definir la longitud de salida de un campo alfanumérico. Si se especifica menor que la longitud del campo, la salida será truncada al tamaño del valor de AL. Cualquier especificación de edición de máscara para el campo será anulada por el valor del parámetro AL.

Posibles valores	Valor por default	Ejemplo
1 al valor del parámetro LS	ninguno	FORMAT AL=20

EM (Editar máscara)

La edición de máscaras da el formato deseado a los campos de salida. Es posible utilizarlas con la sentencia WRITE o DISPLAY como se describió anteriormente. La edición de máscara podrá ser anulada por la especificación de una nueva edición de máscara o por la especificación del parámetro EM=OFF.

Posibles valores	Valor por default	Ejemplo
ON, OFF	Ninguno	EM=ON

ES (Supresión de líneas vacías)

Suprime la impresión de líneas vacías generadas por una sentencia WRITE o DISPLAY . Si se especifica ES=ON, la línea resultante de una sentencia WRITE o DISPLAY que contenga únicamente blancos, no será impresa. Esto es particularmente útil cuando se utilizan arreglos (por ejemplo campos de valor múltiple o campos contenidos dentro de un grupo periódico), para evitar la impresión de un gran número de líneas en blanco.

Posibles valores	Valor por default	Ejemplo
ON / OFF	OFF	FORMAT ES=ON

FC (Carácter de relleno)

El carácter de relleno que aparecerá en un encabezado producido por una sentencia DISPLAY.

Posibles valores	Valor por default	Ejemplo
Cualquier carácter	Blanco	FORMAT FC=*

FL (Longitud de la mantisa de punto flotante)

Define la longitud de salida de la mantisa para una variable de punto flotante. La longitud total es 'FL + 6' para caracteres con signo, exponente y decimal.

Posibles valores	Valor por default	Ejemplo
1 - 16	ninguno	FORMAT FL=5 → +1.2345E+03

HC (Centrado de Encabezado)

Centra o justifica los encabezados de las columnas. Un valor de 'C' indica un centrado normal. Un valor de 'L' causará que el encabezado sea justificado a la izquierda y un valor de 'R' que el encabezado sea justificado a la derecha.

Posibles valores	Valor por default	Ejemplo
C / L / R	C	FORMAT HC=L

IS (Supresión de idénticos)

Suprime la impresión de información idéntica en líneas sucesivas creadas por un WRITE o un DISPLAY. Si se especifica que IS=ON, un valor igual a un valor previo para el mismo campo, no será desplegado. Este parámetro puede ser suspendido para un registro por medio de la sentencia SUSPEND IDENTICAL SUPPRESSS.

Posibles valores	Valor por default	Ejemplo
ON / OFF	OFF	FORMAT IS=ON

LS (Tamaño de la Línea)

Determina el máximo número de caracteres permitidos por línea para un DISPLAY o WRITE.

Posibles valores	Valor por default	Ejemplo
0 - 250	0 (indica que el tamaño de línea del dispositivo de salida va a ser usado)	FORMAT LS=100

MP (Máximo número de páginas)

Determina el número máximo de páginas a ser producidas por un reporte. El valor especificado es el número de páginas físicas y no tiene efecto el número de página de inicio. El programa terminará con un mensaje de error si el valor del MP es excedido.

Posibles valores	Valor por default	Ejemplo
1 - 32767	32767	FORMAT MP=1000

NL (longitud numérica para salida)

Determina la longitud de un campo numérico en una sentencia DISPLAY o WRITE. La longitud es especificada como nn.m donde nn representa el número de posiciones antes del punto decimal y m representa el número de posiciones después del punto decimal. La notación m es opcional y el total de n+m no deberá exceder de 27. El valor de m no podrá exceder de 7.

Si el valor de NL es menor que la longitud del campo el valor será truncado, esto no produce error cuando algunos dígitos relevantes son truncados.

Posibles valores	Valor por default	Ejemplo
1 - 27	27	FORMAT NL=3.2

PS (Tamaño de página)

Indica el número máximo de líneas por página para los reportes creados con las sentencias DISPLAY / WRITE. Un valor de cero para el reporte 0 indica el tamaño de página del dispositivo de salida que será utilizado.

Posibles valores	Valor por default	Ejemplo
0 - 250	0	FORMAT PS=40

SF (Factor de espaciamiento)

Determina el número de espacios a ser insertados entre las columnas de los valores de los campos del reporte creados con la sentencia DISPLAY. Su valor no podrá ser cero, al menos un carácter en blanco deberá ser escrito entre las columnas del reporte.

Posibles valores	Valor por default	Ejemplo
1 - 30	1	FORMAT SF=5

SG (Posición de signo)

Indica si será impreso o no el signo de un campo numérico.

Posibles valores	Valor por default	Ejemplo
ON / OFF	ON	FORMAT SG=OFF

UC (Carácter de subrayado)

Define los caracteres a ser usados para el subrayado de títulos y encabezados. Si no se desea el subrayado, se puede especificar 'UC= ', lo que indica que el carácter de subrayado es blanco.

Posibles valores	Valor por default	Ejemplo
Cualquier carácter	- (guión)	FORMAT UC=*

ZP (Impresión de cero)

Indica si un campo cuyo valor sea cero será impreso o no.

Si el ZP es puesto en 'OFF', cada campo cuyo valor contenga únicamente ceros será suprimido.

Si el ZP es puesto en 'ON', un cero será impreso por cada campo cuyo valor sea ceros.

Posibles valores	Valor por default	Ejemplo
ON / OFF	ON	FORMAT ZP=OFF

Ejemplo de uso de parámetros del FORMAT dentro de un programa

```
0010 DEFINE DATA LOCAL
0020 01  VIEW-ARTICULOS1 VIEW OF ARTICULOS-COMESTIBLES
0030   02  CVE-ART
0040   02  NOMBRE-ART
0050   02  DESCRIP
0060   02  CVE-PRESNT
0070 END-DEFINE
0080 ******************************************
0090 ** INICIO DE PROGRAMA                    **
0100 ******************************************
0110 FORMAT PS = 21 UC = = SF = 3
0120 READ VIEW-ARTICULOS1 BY NOMBRE-ART STARTING FROM 'LECHE'
0130                         ENDING AT 'QUESO '
0140      DISPLAY NOTITLE 1T 'ARTICULO:' NOMBRE-ART (IS=ON)
0150             6X 'CLAVE:' CVE-ART (AL=20)
0160             6X 'DESCRIPCION:' DESCRIP
0170             6X 'PRESENTACION' CVE-PRESNT (AL=30)
0180             SKIP 1
0190 END-READ
0200 ******************************************
0210 ** FIN DE PROGRAMA                       **
0220 ******************************************
0230 END
```

Salida del ejemplo de FORMAT

ARTICULO	CLAVE	DESCRIPCION	PRESENTACION
LECHE	LACLEC3	LECHE ULTRAPASTEURIZ	***ENPAP
	LACLEC5	LECHE EVAPORADA LIGH	***ENLAT
	LACLEC7	LECHE SEMIDESCREMADA	***ENPAP
	LACLEC8	LECHE CONDENSADA PAS	***ENPAP
	LACLEC9	LECHE ADICIONADA CON	***ENLAT
MANTEQUILLA	LACMAN0	MANTEQUILLA CON SAL	***ENPAP
	LACMAN3	MARGARINA	***ENPAP
QUESO	LACQUE2	QUESO CHIHUAHUA	***ENPAP
	LACQUE4	QUESO CREMA PARA UNT	***ENPAP
	LACQUE5	QUESO PARMESANO IMPO	***ENPAP

2.6 Cortes de control

2.6.1 AT BREAK

En la elaboración de reportes es común realizar una serie de operaciones cuando hay un cambio de valor en un campo determinado (cortes) el cual viene ordenado, para lo cual en Natural se cuenta con la sentencia AT BREAK. Esta sentencia se define dentro de un ciclo FIND, READ, HISTOGRAM o READ WORK FILE y ejecuta una o más instrucciones cuando detecta un cambio en el contenido de un campo.

Sintaxis:

[AT] BREAK [(r)] [OF] operando1 [/n/]

sentencias...
END-BREAK

Notación de referencia (r)

Sirve para referenciar el AT BREAK con un especifico ciclo (READ, FIND, etc.). Si (r) es hace referencia a un ciclo de proceso interno, todas las condiciones de corte serán verdaderas cuando este ciclo termine.

Es posible iniciar un nuevo ciclo de procesamiento dentro de una condición AT BREAK. Este ciclo deberá ser también cerrado dentro de la misma condición AT BREAK.

Operando1 (Campo de control)

El campo de control de break es usualmente un campo de la base de datos. Si una variable definida por el usuario es utilizada para este fin, deberá ser inicializada antes de la evaluación del proceso automático del corte, mediante la sentencia BEFORE BREAK PROCESSING, cuya sintaxis es:

BEFORE [BREAK] [PROCESSING]
Sentencias
END-BEFORE

La sentencia BEFORE BREAK PROCESSING es utilizada antes de que:

- el valor del campo de control (variable de usuario) del corte sea evaluado;
- las sentencias especificadas con un AT BREAK sean ejecutadas;
- las funciones de sistema de Natural sean evaluadas.

Ejemplo:

```
0790    ...
0800    READ ARTICULOS BY SP-CVE-ART-PRESNT
0810         BEFORE BREAK PROCCESING
0820                MOVE CVE-ART      TO ÑWA-CVE-ART
0830                MOVE CVE-PRESNT   TO ÑWA-CVE-PRESNT
0840         END-BEFORE
0850         AT BREAK OF ÑWA-SP-CVE-ART-PRESNT
0860            PERFORM CALCULAR-TOTALES
0890            PERFORM IMPRIMIR
0900            RESET ÑWN-SUMA
0900            NEWPAGE
0910         END-BREAK
...     ...
1280    END-READ
...
```

Una ocurrencia específica de un arreglo podrá ser utilizada como campo de control. Un corte ocurre cuando el valor del campo de control cambia o cuando todos los registros en el ciclo para el cual una sentencia AT BREAK es aplicada, han sido procesados.

/n/

La notación '/n/' podrá ser usada para indicar que únicamente las primeras 'n' posiciones del campo de control serán validadas para el cambio en valor.

Niveles de Corte Múltiples

Múltiples sentencias AT BREAK podrán ser especificadas dentro de un ciclo de procesamiento dentro del mismo módulo de un programa. Si se realiza lo anterior, los AT BREAK formarán una jerarquía de niveles de corte, independientemente de si son especificados consecutivamente o intercalados dentro de otras sentencias. El primer AT BREAK representa el más bajo nivel de control de corte y cada AT BREAK adicional representa el siguiente nivel de control de corte más alto.

Junto con esta sentencia se pueden utilizar las siguientes funciones:

Función	Explicación
AVER(campo)	Obtiene el promedio de todos los valores para el campo especificado en los registros procesados
NAVER(campo)	Obtiene el promedio (excluyendo valores nulos) de todos los valores para el campo especificado
MAX(campo)	Obtiene el máximo valor para el campo especificado
MIN(campo)	Obtiene el valor mínimo para el campo especificado
NMIN(campo)	Obtiene el valor mínimo (excluyendo valores nulos) para el campo especificado
OLD(campo)	Obtiene el valor de campo que tenía antes de la ejecución del AT BREAK
SUM(campo)	Obtiene la suma de todos los valores para el campo especificado. SUM es actualizado cada vez que el ciclo en el cual está localizado, es ejecutado. Cuando SUM es seguido por una condición AT BREAK, es reinicializada después de cada valor de break.
TOTAL(campo)	Obtiene el total de todos los valores para el campo especificado. TOTAL no es reinicializado cuando una condición AT se cumple.
COUNT(campo)	COUNT se incrementa en 1 cada vez que se pasa a través del ciclo en el cual está localizado.
NCOUNT(campo)	NCOUNT se incrementa en 1 cada vez que se pasa a través del ciclo en el cual está localizado a menos que el valor del campo especificado sea un valor nulo.

Programa de ejemplo del uso del AT BREAK

```
0010 DEFINE DATA LOCAL
0020 01  VIEW-ARTICULOS1 VIEW OF ARTICULOS-COMESTIBLES
0030    02  CVE-ART
0040    02  NOMBRE-ART
0050    02  CANT
0060    02  CVE-PRESNT
0070 01  ÑWN-SUM-PRES
```

```
0080 01  ÑWN-SUM-NOMBRE
0090 END-DEFINE
0100 ******************************************
0110 ** INICIO DE PROGRAMA                    **
0120 ******************************************
0130 FIND (7) VIEW-ARTICULOS1 WITH NOMBRE-ART='LECHE' OR 'REFRESCO'
0140                                    SORTED BY NOMBRE-ART CVE-
PRESNT
0150    DISPLAY 10X NOMBRE-ART (IS=ON) 10X CVE-ART 10X CVE-PRESNT
15X CANT
0160              ADD CANT TO ÑWN-SUM-PRES
0170              ADD CANT TO ÑWN-SUM-NOMBRE
0180      AT BREAK OF CVE-PRESNT
0190         WRITE NOTITLE 47T OLD (CVE-PRESNT) 67T ÑWN-SUM-PRES /
0200         RESET ÑWN-SUM-PRES
0210      END-BREAK
0220      AT BREAK OF NOMBRE-ART
0230          WRITE NOTITLE 10T OLD(NOMBRE-ART) 67T ÑWN-SUM-NOMBRE
//
0240         RESET ÑWN-SUM-NOMBRE
0250      END-BREAK
0260 END-FIND
0270 ******************************************
0280 ** FIN DE PROGRAMA                       **
0290 ******************************************
0300 END
```

Salida del programa:

```
PAGE1                    12:34:45                          07/11/96

      NOMBRE-ART       CVE-ART       CVE-PRESNT               CANT
      ----------       -------       ----------               -----
      LECHE            LACLEC5          ENLAT                 130
                       LACLEC9          ENLAT                  70
                                        ENLAT                 200

                       LACLEC3          ENPAP                  75
                       LACLEC7          ENPAP                 225
                       LACLEC8          ENPAP                 100
                                        ENPAP                 400

      LECHE                                                   600


      REFRESCO         BEBJUM1          ENLAT                 320
                       BEBPEPL          ENLAT                 180
                                        ENLAT                 500

                       BEBCOC2          ENVAS                 240
                       BEBPAU0          ENVAS                 110
                                        ENVAS                 350

      REFRESCO                                                850
```

2.6.2 AT END OF DATA

En esta sentencia se declaran las instrucciones a ser ejecutadas cuando todos los registros de una base de datos seleccionados por un ciclo de procesamiento han sido procesados. Deberá ser especificado dentro del mismo módulo del programa que contenga el ciclo. Se utiliza únicamente con los ciclos FIND, READ, READ WORK FILE, HISTOGRAM o SORT. Podrá utilizarse solamente un AT END OF DATA por ciclo.
Sintaxis:

[AT] END [OF] DATA [(r)]
Sentencias
END-ENDDATA

Referencia para especificar el ciclo de procesamiento (r):
Esta sentencia podrá ser relacionada a un ciclo especifico de procesamiento con la notación (r). Si no es utilizada, será relacionada al ciclo de procesamiento activo más externo.

Valores de los campos de la base de datos:
Cuando una condición AT END OF DATA ocurre para un ciclo, todos los campos de la base de datos contienen los datos del último registro procesado.

Junto con esta sentencia se pueden utilizar las funciones explicadas en el punto anterior.

2.6.3 AT END OF PAGE

Esta instrucción contiene los procesos que se ejecutarán cuando una condición de fin de página es detectada como resultado de un SKIP o NEWPAGE, pero no como resultado de una sentencia EJECT. La condición de fin de página es revisada después de que una sentencia DISPLAY o WRITE es completada, por lo que si un DISPLAY o un WRITE produce múltiples líneas de salida, un desbordamiento del tamaño físico de la página podrá ocurrir antes de que la condición de fin de página sea detectada.

Sintaxis:

[AT] END [OF] PAGE [(rep)]
sentencias
END-ENDPAGE

Especificación de reporte (rep):
Indica el número del reporte para el cual se aplica la sentencia AT END OF PAGE. Su valor debe estar comprendido entre 1 y 31, o bien deberá ser un nombre lógico el cual ha sido asignado previamente por medio de la sentencia DEFINE PRINTER. Si (rep) es omitido, el valor será el del primer reporte (0).

Tamaño Lógico de la página:

Deberá proporcionarse un tamaño lógico de página menor al tamaño físico de la misma para asegurar que la información impresa por un AT END OF PAGE aparezca en la misma página física.

Control de la última página:

Dentro de un programa principal, una condición de fin de página es activada cuando la ejecución del programa principal termina vía ESCAPE, RETURN o END.

Dentro de una subrutina, una condición de fin de página no es activada cuando la ejecución de una subrutina termina vía ESCAPE, RETURN o END.

Puede utilizar la mismas funciones que se emplean junto con el AT BREAK.

2.6.4 AT START OF DATA

Es utilizada para desarrollar procesos inmediatamente después de que el primero de un conjunto de registros es leído por ciclo iniciado por un READ, FIND, HISTOGRAM, SORT o READ WORK FILE. Si el ciclo de inicio contiene una cláusula WHERE, la condición de inicio de datos será verdadera cuando el criterio de búsqueda básico y el del WHERE se cumplan. Esta sentencia podrá ser usada únicamente una vez dentro de un ciclo dado.

Sintaxis:

[AT] START [OF] DATA [(r)]
sentencias
END-START

Valor de los campos de la base de datos:

Todos los campos de la base de datos contienen los valores del registro el cual causó que fuera verdadera la condición del inicio de los datos.

Referencia para especificar el ciclo de procesamiento (r)

Una sentencia AT START OF DATA podrá ser relacionada a un ciclo determinado, empleando la notación '(r)'. Si no se especifica se relacionará al ciclo activo más externo.

2.6.5 AT TOP OF PAGE

Esta sentencia contiene las instrucciones a ser ejecutadas cuando una nueva página es iniciada. Esto ocurre cuando el contador interno de líneas excede el parámetro de tamaño de página (PS) o cuando un NEWPAGE es ejecutado. La sentencia EJECT inicia una nueva página pero no se cumple la condición al principio de la página. Cualquier salida creada como resultado de un AT TOP OF PAGE aparecerá en seguida de la línea de título con una línea intermedia en blanco.

Sintaxis:

[AT] TOP [OF] PAGE [(REP)]
sentencias
END-TOPPAGE

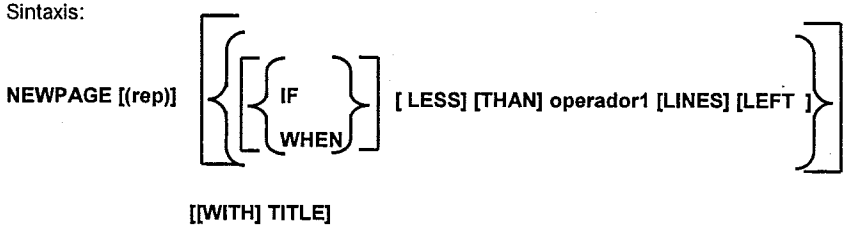
Especificación de reporte (rep)

La notación (rep) podrá ser usada para especificar la identificación del reporte para el cual el AT TOP OF PAGE es aplicable. Su valor debe estar comprendido entre 1 y 31, o bien escribir un nombre lógico el cual debe ser asignado previamente con la sentencia DEFINE PRINTER. Si no se especifica, el valor será el del primer reporte (0).

2.6.6 NEWPAGE

NEWPAGE avanza a una nueva página. Esta instrucción hace verdaderas las condiciones de fin de página y de write-trailer. Si es omitido, el avance de página es controlado automáticamente basándose en el parámetro PS.

Sintaxis:



Especificación de reporte (rep):

(rep) es la identificación del reporte para el cual el NEWPAGE es aplicado. Su valor debe estar comprendido entre 1 y 31, o bien ser un nombre lógico asignado previamente mediante la sentencia DEFINE PRINTER. Si (rep) es omitido, el valor será el del primer reporte (0).

Opciones especiales:

EVEN IF TOP: Causa que se genere una nueva página, aún si ésta fue iniciada inmediatamente antes de que el NEWPAGE fuera encontrado.

WHEN LESS THAN LINES LEFT: Causa que se genere una nueva página cuando el valor del operando1 es menor al número de líneas totales.

WITH TITLE: Especifica un título el cual será escrito en la nueva página. El título es especificado empleando la misma sintaxis como se describió para el WRITE TITLE.

3. EDITORES

Menú principal de Natural

El menú principal de NATURAL se presenta inmediatamente después de haber capturado la librería, el usuario y el password correspondientes.

Este menú presenta varias opciones y es el punto de partida para el desarrollo de las aplicaciones NATURAL. Para estas aplicaciones es necesario editar programas, áreas de trabajo y pantallas.

La siguiente figura presenta este menú:

```
16:58:12 *** NATURAL APPLICATION DEVELOPMENT SYSTEM *** 19/03/97
User: SINERS01          - Main   Menu -          Library: VENTAS
                                                       Mode Structured

                  code    Function
                  ----    --------------------------
                  D       Development Facilities
                  R       Reporting Facilities
                  A       Administration Facilities
                  ?       Help
                  .       Terminate Natural Session
                  ----    --------------------------
                           Enter Code:     DMode:


        Command:

Enter-PF1--PF2--PF3--PF4--PF5--PF6--PF7--PF8--PF9--PF10--PF11--PF12
       HELP ---  EXIT  D    R    A  ---  ----  ---- ----  ----  ----
```

La opción D (Development Facilities) permite al usuario accesar el ambiente para desarrollar aplicaciones.

Para construir los diferentes elementos de un sistema, se necesitan de tres tipos de editores:

- Editor de programas
- Editor de áreas de trabajo
- Editor de pantallas (mapas)

3.1 Editor de programas.

```
16:58:12 *** NATURAL APPLICATION DEVELOPMENT SYSTEM *** 19/03/97
User: SINERS01      - Development  Facilities -   Library: VENTAS
                                                       Mode: Structured
                  Code     Function
                  ----     ----------------------------------------
                  C        Create object
                  E        Edit object
                  R        Rename object
                  R        Delete object
                  X        Execute program
                  L        List  Objects, X-Ref
                  G        Global Environment
                  ?        Development Facilities Help
                  .        Terminate Development  Facilities
                  -----    ----------------------------------------
                  Enter Code:      E     Type:  P  Name: PGMART01

Command:

Enter-PF1--PF2--PF3--PF4--PF5--PF6--PF7--PF8--PF9--PF10--PF11--PF12

      HELP MAIN EXIT C     E    R    D    X    T     L    G     --
```

El editor de programas se invoca con el comando EDIT, seguido del tipo de objeto que se va editar, en la línea de comando (Command:) o como se muestra en la figura anterior, con el código E (Enter Code), el tipo de objeto P (Type) y el nombre del objeto PGMART01 (Name).

Los siguientes son los tipos de objeto que se pueden editar.

Tipo-objeto

EDIT { COPYCODE | DESCRIPTION | HELPROUTINE | PROGRAM | SUBPROGRAM (N) | SUBROUTINE } nombre-objeto [librería]

Tipo de objeto

COPYCODE : Parte de código fuente de NATURAL el cual puede ser incluido en programas de Natural usando el postulado INCLUDE, este código se agrega en tiempo de compilación.

DESCRIPTION: Descripción del programa, se especifica solo si se encuentra instalado PREDICT.

HELPROUTINE: Rutina que va a ser invocada para proporcionar información de ayuda. Esta rutina se puede codificar en el editor de programas de NATURAL o en el editor de pantallas.

PROGRAM: Programa principal de NATURAL.
SUBPROGRAM: Es un subprograma que se invoca mediante el postulado CALLNAT.

SUBROUTINE: Rutina externa que se invoca mediante el postulado PERFORM.

Para editar un objeto se necesita definir su tipo y se puede hacer especificando sólo la letra subrayada o todo el nombre del tipo de objeto.

Ejemplos :

```
Command:    EDIT PROGRAM  object-name
Command:    E P  object-name

Command:    EDIT COPYCODE object-name
Command:    E C  object-name
```

Nombre del objeto

Si se proporciona el nombre del objeto, NATURAL cargará en el área de edición de trabajo dicho objeto y pondrá el nombre para posterior utilización de los comandos SAVE, CATALOG o STOW.
Ejemplo:

```
Command:    E P  PGMART01
```

```
>                    > +   Program  :  PGMART01  Lib:  VENTAS
All      ...+..1..+..2..+..3..+..4..+..5..+.. MODE   : Struct ..
0010 DEFINE DATA
0020 GLOBAL USING GLOART01
0030 LOCAL  USING LOCART01 /* VIEW-ARTICULOS vista de ARTICULOS
                                                      COMESTIBLES
0040 LOCAL
0050 ÑCVE-ARTICULO              (A7)  INIT <'LACMAN1>
0060 END-DEFINE
0070 ********************************
0080 **  inicio del programa       **
0090 ********************************
0100 READ VIEW-ARTICULOS BY CVE-ART STARTING FROM ÑCVE-ARTICULO
0110      DISPLAY CVE-ART  CVE-PRESNT DESCRIP CANT
0120 END-READ
0130 ********************************
0140 **  fin del programa          **
0150 ********************************
0160 END
0170
 .+.+. Current Source Size : 308 Char . Free: 28924 +. $ 16 L 1
```

Por otra parte si no se proporciona el nombre del objeto se presentará el área de trabajo vacío, con el propósito de capturar el código del objeto en cuestión.

Ejemplo:

Command: E P

```
>                    > +     Program    :          Lib: VENTAS
All ..+..1..+..2..+..3..+..4..+..5..+..        MODE : Struct  ..
0010
0020
0030
0040
0050
0060
0070
0080
0090
0100
0110
0120
0130
0140
0150
0160
0170
0180
.+.+..Current Source Size : 0  Char . Free: 28924 + . . . S 1 L 1,
```

nombre de la librería

La librería no necesita especificarse, ya que esta ha sido proporcionada en el momento de ingresar al sistema y además únicamente se pueden accesar las librerías que están ligadas con el usuario de NATURAL. En el caso en que se tenga acceso a varias librerías se puede cambiar de librería con el comando LOGON (LOGON library-id).

El editor de programas acepta dos tipos de comando: Comandos de edición y comandos de línea. Los comandos de edición se capturan en la línea de comando (>) y los comandos de línea se capturan en la primer columna en la línea en que se quiera realizar alguna actividad.

Comandos de edición:

- ADD — Agrega a la pantalla actual 9 líneas vacías. Si el indicador de dirección es '+', las líneas se agregan después de la última línea; si el indicador de dirección es '-', las líneas vacías se agregan antes de la primera línea de la pantalla actual.
- CHANGE 'scandata' 'replacedata' — Reemplaza el valor buscado 'scandata' por el valor 'replacedata'.
- CLEAR — Limpia el área de trabajo.
- DX, DY, DX-Y — Borra la línea marcada con X o Y, o borra el bloque que empieza desde la línea marcada con X hasta la línea marcada con Y.
- EX, EY, EX-Y — Borra las líneas de código desde la primera línea hasta una línea antes de la línea marcada con X, o las líneas que siguen después de la línea marcada con Y, o las líneas que están antes y después del bloque que forman las líneas marcadas con X y Y.
- N — Renumera el programa actual en el área de trabajo.
- POINT — La línea en la cual se capturó el comando de línea '.P' se posiciona en el tope de la pantalla .
- PROFILE — Edita el PROFILE actual.
- RESET — Borra las marcas X o Y, además la marca que se hizo con el comando '.P'.
- SCAN ['scan-value'] — Busca o rastrea el valor 'scan-value' en el área fuente. Si no se da el valor, se presenta el menú SCAN en el cual se captura el valor a rastrear. La búsqueda o rastreo se determina por el indicador de dirección (+ , -). Si el indicador de dirección es '+', la búsqueda se hará desde el tope de la pantalla actual hasta la última línea del área fuente. Si el indicador de dirección es '-', la búsqueda se hará de la posición actual (sin incluir la línea actual) a la primera línea del área fuente.
- SET ABS ON/OFF — SET ABS ON opera en modo absoluto; es decir el valor a rastrear no necesita estar limitado por caracteres.
- SET ESC carácter — Define el carácter de escape. El carácter de escape por default es ' . '.
- SET RANGE ON/OFF. — Con SET RANGE ON los comandos SCAN y CHANGE únicamente se efectuarán en el bloque delimitado por las líneas marcadas con X y Y.
- SET NUL ON/OFF — SET NUL ON causa que el valor buscado por el comando SCAN sea reemplazado por un valor nulo.
- SET SIZE ON/OFF — SET SIZE ON muestra el tamaño del programa en la parte baja de la pantalla de edición.
- SET STAY ON/OFF — STAY=ON para avanzar/regresar una pantalla solo se puede hacer con los comandos de posición. Si STAY=OFF con el ENTER se podrá avanzar una pantalla.
- SET TYPE — Este comando se utiliza para cambiar el tipo de objeto que se está editando.
- SHIFT [-/+nn] — Desplaza cada línea delimitada por las marcas X y Y a la izquierda (-) o a la ha(+) el número de caracteres especificado (nn). derec
- STRUCT — Realiza la identación estructurada del código fuente.

- *= Lleva a cabo la ejecución del último mandato capturado en la línea de comandos.

Los siguientes *comandos de posición* también se editan en la línea de comandos:

- +P, -P Avanza o regresa una página
- +, - Avanza o regresa una página
- +H,-H Avanza o regresa media página
- -- Se posiciona en el principio del programa
- ++ Se posiciona en el final del programa
- -nnn Avanza o regresa un número (nnn) de líneas
- nnnn Se posiciona en la línea nnnn
- X, Y Se posiciona en la línea que tiene la marca X o la Y

Comandos de línea

Nota: (nn) indica el número de repeticiones. El valor por default es 1, solo para el comando .I el valor de default es 9 o 4 en el caso de pantalla compartida).

- .C(nn) Copia la línea en la cual se captura este comando.
- .CX(nn) .CY(nn) Copia la línea marcada con X o la línea marcada con Y.
- .CX-Y(nn) Copia el bloque de líneas delimitado por la X y la Y.
- .D(nn) Borra n líneas.
- .I(nn) Inserta (nn) líneas o 9 líneas por default. Si el indicador de dirección es '+', inserta las líneas después de la línea actual. Si el indicador de dirección es '-', inserta las líneas antes de la línea actual.
- .I(pgm,sss,nn) Inserta un programa fuente que reside en la librería actual o en la librería SYSTEM. 'sss' se usa para indicar el número de línea donde empieza el código del programa que se desea insertar y 'nn' es el número de líneas que se van a insertar.
- .J Junta la línea actual con la siguiente línea. Si al juntar las dos líneas la longitud es mayor que la longitud de la pantalla de edición, la línea será marcada con una 'L' y debe ser separada usando el comando .S (Split).
- .L Al ejecutar este comando la línea permanece sin cambio.
- .MX .MY Mueve la línea marcada con X o la marcada con Y.
- .MX-Y Mueve el bloque delimitado por las líneas marcadas con X y Y.
- .N Marca una línea como referencia para el comando '.P' .
- .P Posiciona la línea referenciada por el comando '.N' en el tope de la pantalla.
- .S Separa la línea en la posición en que se encuentre el cursor.
- .X Marca el comienzo de un bloque de líneas a ser procesadas.
- .Y Marca el fin de un bloque de líneas a ser procesadas.

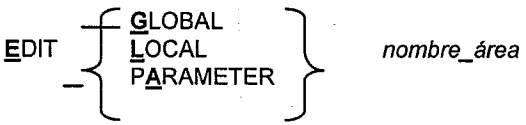
3.2. Editor de áreas de trabajo.

Usado para definir y dar mantenimiento a las áreas globales, locales y de parámetros. En este editor se definen las variables de usuario, vistas de archivos de la base de datos o bloques de datos.

Esta área de trabajo se puede invocar desde la pantalla Development Facilities capturando la opción E (Edit object), el tipo de objeto que se desea editar (G, L, A) y el nombre del objeto en caso de que exista. También se puede invocar desde la línea de comandos.

Los objetos que se pueden editar en esta área de trabajo son los siguientes:

tipo_objeto



Tipo de objeto

- GLOBAL: área de datos globales, se definen las variables que se utilizan en toda la aplicación.
- LOCAL: área de datos locales, se definen las variables o vistas utilizadas únicamente por el programa.
- PARAMETER: edición de parámetros de comunicación entre programas y subprogramas.

Nombre del área de trabajo

Si se da el nombre del área de trabajo se presentará el editor de áreas de trabajo con el objeto especificado. Si no se captura el nombre, el editor se presentará vacio.

```
16:58:12 *** NATURAL APPLICATION DEVELOPMENT SYSTEM  ***  19/03/97
User: SINERS01       Development  Facilities -    Library: VENTAS


              Code   Function
              ------ ----------------------------------------
              C      Create object
              E      Edit object
              R      Rename object
              D      Delete object
              X      Execute program
              L      List  Objects, X-Ref
              G      Global Environment
              ?      Development Facilities Help
              .      Terminate Development  Facilities
              ------ ----------------------------------------
      Enter Code:        E       Type : L Name: LOCART01

  Command:
Enter-PF1-PF2-PF3-PF4-PF5-PF6-PF7-PF8-PF9-PF10-PF11-PF12-
      HELP MAIN EXIT C E   R   D   X   T    L    G  ----
```

Ejemplo de edición de una local existente :

Command: E L LOCART01

```
16:27:33      *****    EDIT DATA      *****              10-03-97
Library : VENTAS      Name : LOCART01   DBID : 001   FMR:       5
Command :                                                  >  +
I  T  L   Name      __      F  Leng   Index /Init /EM/Name/Comment
-- -- -- -------- all --- -- -----   ---------------------------
       1 VIEW-ARTICULOS     /* vista de ARTICULOS-COMESTIBLES
       2  CVE-ART             A   7
       2  NOMBRE-ART          A   10
       2  CVE-PRESNT          A   5
       2  DESCRIP             A   30
       2  CANT                N   5
```

Ejemplo de edición de una local nueva:

Command: C L

Presentará el área de trabajo vacía.

```
16:27:33            *****   EDIT DATA     *****          20-03-97
Library : VENTAS       Name :              DBID : 001  FMR:       5
Command :                                                  >  +
I  T  L     Name      __        F  Leng  Index /Init /EM/Name/Comment
-- -- --  --------- all ---  -  ----   ---------------------------
```

Los campos de la pantalla anterior funcionan de la siguiente manera:

- I Códigos de información que proporciona el editor :
  E indica que un error de definición ha sido detectado
  I indica que un valor ha sido definido
  M una máscara ha sido definida
  S un valor inicial y una máscara han sido definidos
  Esta columna no la puede modificar el usuario.
- T Tipo
  V vista
  G grupo
  M múltiple
  P periódico
  R redefinición de campos
  C constante
  * comentario
- L Nivel de anidamiento
- NAME Nombre de la variable, del bloque o de la vista
- F Formato de la variable (alfanumérico, numérico, etc.)
- LENG Longitud del campo. No se puede capturar para los siguientes formatos (C,D,T)
- INDEX/... Se definen los arreglos, se dan valores iniciales a los campos, para editar máscaras, para escribir el nombre de los archivos de los cuales se tomaron las vistas, nombres de bloques, comentarios.

Este editor soporta los siguientes comandos de edición:

- CHECK Checa el área de datos fuente actualmente cargado en el editor.
- CLEAR Limpia el área fuente de datos.
- CREATE GLOBALS Junta todas las variables globales que están en la librería actual y las ubica en el área de datos global denominada COMMON.
- GEN object-name Genera un copycode de las definiciones que están en el área de trabajo actual dentro del DEFINE DATA LOCAL.
- READ data-area Lee una definición de área de datos existente dentro del área fuente.
- SAVE data-area Salva el área de datos que se encuentra en el área fuente, sin revisar las definiciones de las variables.
- SET STAY ON/OFF Solo se pueden usar los comandos de posición para avanzar/regresar páginas si se encuentra en STAY=ON. Si está en STAY=OFF se podrá avanzar una página con el ENTER.
- SET TYPE Puede cambiarse el tipo de objeto mediante este comando; G cambia a global; L a local y P a parámetros.
- STOW data-area Salva y cataloga los datos que están actualmente en el área fuente, las definiciones son revisadas antes de ser catalogadas.

Los siguientes son comandos de línea :

- .E Invoca una pantalla separada para definir los valores iniciales y editar las máscaras.
- .R Se utiliza para redefinir una variable.
- .V [(view-name)] Define una vista. Se presenta el layout de la vista como está definida, el usuario debe seleccionar los campos de la vista. Si no se especifica el nombre de la vista, se generará con el mismo nombre con el que ha sido declarada.
- .* Genera un contador interno de ocurrencias (C*) , para campos múltiples o campos periódicos.
- number Solo en modo pantalla compartida. Sirve para obtener un campo de la pantalla compartida especificando el número del campo en la primer columna. Este campo se insertará antes o después dependiendo del indicador de dirección.

3.3 Editor de pantallas.

Para crear una pantalla se llama al editor de pantallas(mapas) mediante el comando EDIT (E M MAPSIN01) o desde la pantalla Development Facilities como se muestra en la siguiente figura.

Se utiliza para crear pantallas interactivas o formas para reportes. Las pantallas se invocan con el postulado INPUT desde un programa mientras una forma se invoca con un WRITE.

```
16:58:12 *** NATURAL APPLICATION  DEVELOPMENT SYSTEM *** 19/03/97
User:  SINERS01   - Development Facilities -      Library: VENTAS
                                                    Mode : Stuctured

                    Code   Function
                    -----  ------------------------------------
                    C      Create object
                    E      Edit object
                    R      Rename object
                    D      Delete object
                    X      Execute program
                    L      List  Objects, X-Ref
                    G      Global Environment
                    ?      Development Facilities Help
                    .      Terminate Development  Facilities
                    -----  ------------------------------------
       Enter Code:  E  Type : M Name: MAPSIN01

      Command:
Enter-PF1--PF2--PF3-PF4-PF5-PF6-PF7--PF8--PF9--PF10--PF11--PF12--
       HELP MAIN EXIT C  E   R   D    X   T     L     G   ---
```

Para crear una pantalla hay que realizar lo siguiente:

a) Definir el profile de la pantalla(tamaño de la pantalla, delimitadores de campo, caracteres de relleno a ser usados, etc.) . Se presenta un menú para definir este profile. Si no se quiere realizar ningún cambio se puede dejar el profile con los valores que tiene por default.

b) Definir los campos que estarán en la pantalla mediante el Display Field Definitions o directamente en la pantalla. Para la definición de campos puede ayudarse con los comandos de posición, de línea, de campo y las facilidades del editor.

c) Salvar y catalogar la pantalla (mapa) para que se pueda invocar desde un programa de Natural.

```
12:27:33                                                  20-03-97
User :SINERS01          -- Edit Map --

      Code      Function                    Parameter
      -----     --------------------------  --------------
      D Display Field  Definitions                Name. Profile
      E Edit Map                            Name. Profile
      I Initialize new Map                        Name. Profile
      H Initialize a new Help Text                Name. Profile
      M Maintenance of Profiles & Devices   Profile
      S Save Map                            Name. Profile
      T Test  Map                                 Name. Profile
      W Stow Map                            Name. Profile
      ? Help
      . Terminate
 Select Code:  Name:
                                                  Profile: SYSPROF
                                                   Library: COURSES

 or  Direct Command:
Enter-PF1--PF2--PF3---PF4-PF5-PF6-PF7-PF8--PF9--PF10--PF11--PF12--
     HELP       Quit  Test Edit
```

La siguiente información explica cada uno de los códigos del menú de edición de mapas.

- Name — Nombre del mapa
- Profile — Parámetros de la sesión actual
- Library — Librería actual en la cual se salvará la pantalla
- D — Despliega el nombre, formato, modo, posición y número de reglas de proceso para cada campo
- E — Invoca el editor de pantallas para crear o modificar un mapa
- I — Inicializar un nuevo mapa
- H — Inicializar una nueva ayuda de texto
- M — Modificar el profile (características del mapa)
- S — Salvar la definición del mapa (pantalla)
- T — Prueba de la pantalla, para asegurarse que funcione satisfactoriamente; incluye la prueba de las reglas de proceso y las facilidades de ayuda
- W — Cataloga la pantalla y la salva

Inicialización de una pantalla

Esta sección describe los procesos de modificación de las características (profile) de la pantalla.

Delimitadores:

Son usados como prefijo para el campo o el texto constante para indicar el tipo de campo, atributo y color.

- Tipos de campo (Cls):

  A campo de entrada
  M campo modificable (salida/entrada)
  O campo de salida no modificable
  T texto constante

```
09:36:47                                          Define Map Settings for
MAP       26/03/97
-------------------------------------------------------------------
Delimiters:          Format:                        Context:
--------------       -------------------   ----------------------
Cls  Atr CO Del      Pagesize    25       Device Check
T    D      BLINK    Linesize    78       Write Stant
T    I      ?          Layout                      Input Stant X
A    D                Zero Print  N (Y/N)    Help
A    I      )         Case Deflt  UC (UC/LC) as field default N (Y/N)
M    D      *         Decimal Ch         Profile Name SYSPROF
M    I      :         Std Keys       N (Y/N)
O    D      +         Justified      L (L/R)
O    I      (         Print Mode         Filler  Characters:
                                                  ----------------------
                        Control   Var                 Optional, Partial
                                                      Required, Partial
                                                      Optional, Complt
                                                      Esquired, Complt

Enter --PF1--PF2-- PF3--PF4-PF5-PF6-PF7-PF8-PF9-PF10-PF11-PF12---
       HELP  ---   Quit --  --- --- --- --- --- ---  ---    Let
```

- Atributos (Atr):
  B blinking (parpadeante)
  C cursiva/itálica
  D default (no intenso, no blinking, etc)
  I intenso
  N no desplegable
  U subrayado
  V video inverso
  Y dinámico (los atributos se asignan dinámicamente)

- Colores (Co):
  BL azul
  GR verde
  EN neutro (blanco)
  PI rosa
  RE rojo
  TU turquesa
  YE amarillo

- Delimitadores (del):

  Son caracteres especiales usados en el mapa que sirven para delimitar o indicar (al inicio) el tipo de campo y sus atributos.

  Los caracteres de default son los siguientes:

  blink ?) * : + (

Formato:

Page Size	Número de líneas de la pantalla (1 a 250), si se tienen que presentar las PF's (3 a 250).
Line Size	Número de columnas de la pantalla (5 a 249).
Layout	El nombre de un mapa fuente definido que contiene un layout predefinido.
Zero Print	Con "Y" pone los ceros de la izquierda de los campos numéricos. Con "N" los ceros de la izquierda de los campos numéricos se representan con blancos.
Case Deflt	UC indica que las letras de un campo alfanumérico que sean minúsculas serán convertidas a mayúsculas. LC No se hace ninguna conversión de minúsculas a mayúsculas.
Manual Skip	Si se pone " Y " el cursor tendrá que moverse manualmente al próximo campo después de haber capturado la información en el campo actual.
Decimal Ch	El caracter a ser usado como decimal.
Std Keys	Con "Y" deja las dos últimas líneas del mapa con el propósito de especificar las teclas estándar de funciones en tiempo de ejecución.
Justified	Tipo de justificación del campo (R justificado a la derecha, L justificado a la izquierda).
PM	El modo de impresión para variables: *C* Indica poner un caracter alternativo para ser usado (caracter especial) *I* Indica dirección de impresión inverso *N* Permite desplegar en pantalla, pero no se podrá imprimir vía el comando %H
Control Var	El nombre de una variable de control.

- **Contexto:**

Device Check	Si el nombre de un dispositivo se captura, el mapa validará la compatibilidad con el profile del dispositivo.
Write Statement	Si se captura un valor no blanco se indica que el mapa se invocará mediante una sentencia WRITE.
INPUT Statement	Si se captura un valor no blanco se indica que el mapa se invocará mediante una sentencia INPUT.
Help Routine	Se captura el nombre de una rutina de ayuda a ser invocada en tiempo de ejecución.
As Field default	Si se define "Y", la Helproutine especificada para el mapa se aplicará para cada campo individual del mapa, es decir el nombre de cada campo es pasado individualmente a la rutina de ayuda. En otro caso el nombre del mapa es pasado a la Helproutine.
Profile Name	El nombre del profile de la sesión el cual inicializará el mapa. Si ENFORCED es desplegado, únicamente se pueden modificar las siguientes características: Helproutine, Upper/lower case conversion, field justification, presentacion mode y dynamic attribute.

- **Filler Caracters:**

Los caracteres de relleno pueden ser asignados para indicarle al operador de la terminal si la información es requerida y si los campos deben ser llenados completamente.

Requerimiento	Explicación
Optional, Partial	Si no es requerido no necesita ser llenado completamente.
Required, Partial	Requerido, no necesita ser llenado completamente(AD=E).
Optional, Complete	No es requerido, debe ser llenado completamente(AD=G).
Required, Complete	Requerido, debe ser llenado completamente(AD=EG).

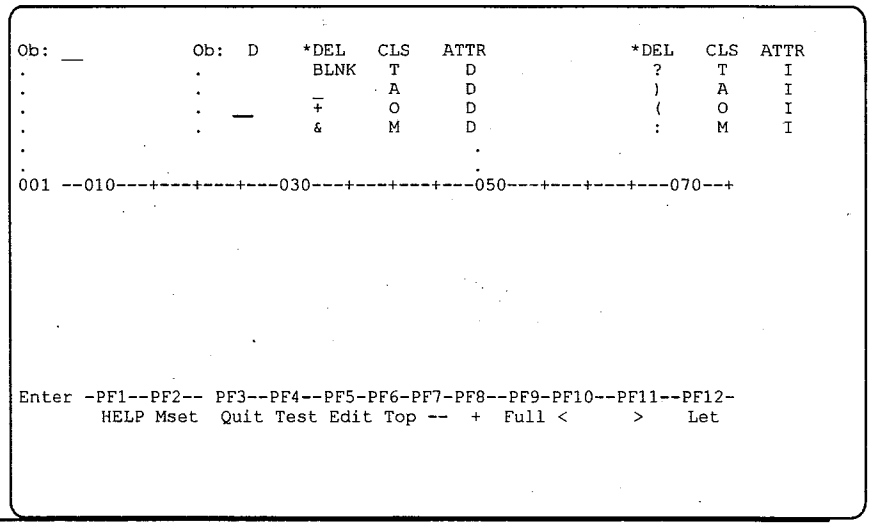
Edición de mapas

El editor de pantallas también tiene el modo de pantalla compartida. La parte superior de la pantalla es usada para definiciones de vistas de usuario y la parte de abajo para definición del mapa; la tecla de función <PF9> cambia de pantalla completa a pantalla compartida.

Si se captura un "." en la primera posición del campo de la esquina superior izquierda (Ob: __), causará que regrese al menú principal.

Teclas y comandos de posición

PF key	Comando	Explicación
PF1		Invoca la ayuda del mapa
PF2		Conjunto de características del mapa
PF3	.Q	Regresa al menú principal
PF4		Prueba de la definición del mapa
PF5	.E	Invoca la edición del campo en el cual el cursor se posicionó
PF6	.--	Regresa al tope de la pantalla
PF7	.-	Mueve la pantalla hacia arriba media página
	.-nnn	Sube el número de líneas especificado por nnn
PF8	.+	Mueve media pantalla hacia abajo
	.+nnn	Baja el número de líneas especificado por nnn
	.++	Se va hasta la parte baja de la pantalla
PF9	./	Cambia de pantalla completa a pantalla compartida
PF10	.<	Desplaza media página a la izquierda
	.<nnn	Desplaza el número (nnn) de columnas a la izquierda
	.<<	Desplaza toda la pantalla a la izquierda
PF11	.>	Desplaza media página a la derecha
	.>nnn	Desplaza el número (nnn) de columnas a la derecha
	.>>	Desplaza toda la pantalla a la derecha
PF12		Ignora los cambios hechos en la pantalla hasta el uso del último ENTER
	.*	Mueve la pantalla a la posición del cursor



Comandos de línea

Comando	Explicación
..A	Definición de una tabla.
..An	Definición de una tabla con n ocurrencias.
..C	Para centrar la línea en donde se captura este comando. Si se escriben dos comandos formando un bloque, se centrarán todas las líneas que formen este bloque.
..Cn	Centrar la línea y mover las n-1 líneas de abajo de acuerdo a ésta.
..C*	Centrar la línea y mover todas las líneas de abajo de acuerdo a ésta.
..D	Borra la línea actual. Si se capturan dos comandos ..D en forma de bloque se borrarán todas las líneas que forman el bloque.
..Dn	Borra la línea y las n-1 líneas que le siguen.
..D*	Borra la línea y todas las líneas que le siguen.
..E	Invoca la edición de campo extendido para todos los campos de la línea actual. De igual forma se pueden dar dos comandos para formar un bloque.
..En	Invoca la extensión de edición de campos para la línea y las n-1 líneas que le siguen.
..E*	Invoca la extensión de edición de campos para la línea y las líneas que le siguen.
..I	Inserta una línea. La última línea vacía es borrada para insertar la línea.
..In	Inserta n líneas abajo de donde se capturó el comando.
..I*	Inserta tantas líneas como sea posible abajo de la línea en la que se capturó el comando.
..Fc	Llena los espacios vacíos de una línea con el caracter 'c'.
..J	Junta la línea actual en la que se captura el comando con la línea que le sigue.
..J*	Une la línea actual con todas las que le siguen excepto para la última línea. Si al realizar la operación Join, la línea resulta ser más grande que la línea definida en el profile, debe utilizarse el comando Split para separar la línea.
..M	Mueve la línea en la que se captura el comando a la posición del cursor. Si dos comandos se capturan formando un bloque, este bloque se moverá a la posición del cursor.
..Mn	Mueve la línea actual y las líneas n-1 que le siguen a la línea marcada con el cursor.
..M*	Mueve la línea actual y las líneas que le siguen tantas como sea posible a la posición del cursor.
..P	Invoca la edición de una regla de proceso para una PF-key.
..Pn	Invoca la edición de reglas de proceso para PF-key con rango 'n'.
..Q	Retorna al menú principal.
..R	Repite una vez todos los textos constantes de la línea actual. Si se forma un bloque con dos comandos de repetición, todos los textos constantes que estén dentro del bloque se repetirán.
..Rn	Repite todos los textos constantes de la línea actual y de las líneas n-1 siguientes.
..S	Separa la línea a partir de la posición del cursor. Si dos comandos delimitan un bloque de líneas, éstas se separarán.
..Sn	Separa la línea en donde el comando es capturado y las n-1 líneas que le siguen en la posición del cursor.

Comandos de campo

Estos comandos se capturan en la primer columna en donde empieza la definición del campo.

Comando	Explicación
.A	Definición de un arreglo. Este comando se aplica a un campo únicamente.
.An	Definición de un arreglo con n ocurrencias.
.C	Para centrar un campo o un rango de campos entre campos adjuntos.
.D	Borra el campo o rango de campos en donde se capture el comando. Si un elemento de un arreglo es borrado todo el arreglo se eliminará.
.E	Invoca la extensión de edición de campo para el campo en donde se capture el comando. Si se posiciona el cursor en el campo seleccionado y se presiona la tecla de función <PF5>, se tendrá el mismo efecto.
.J	Junta los campos localizados en líneas consecutivas.
.M	Mueve el campo o rango de campos a la posición del cursor. Si se quiere mover un rango, se captura un comando en el primer campo que se desea mover y otro en el último, posteriormente se posiciona el cursor en donde se quiere mover el bloque.
.P[nn]	Invoca la edición de una regla de proceso para un campo. Se puede dar un parámetro (nn) para indicar el rango de la regla de proceso.
.R	Repite o copia un campo o un rango de campos. La repetición siempre se hará hacia abajo y de izquierda a derecha. Los campos generados por este comando se les asigna un nombre modelo, por lo que se le debe asignar un nombre válido usando el comando .E .
.S	Separa una línea o un rango de líneas a partir del campo en donde se capturó el comando.
.T	Trunca (borra) un campo o un rango de campos de una línea. Esta función es usada para truncar (borrar) un elemento de un arreglo, el arreglo entero será borrado.

Definición de campos de pantalla (mapa)

Los datos se pueden definir de las siguientes formas:
- Definirlos directamente en la pantalla
- Se pueden seleccionar de una vista de usuario o de una definición de datos
- Se pueden utilizar variables del sistema

Definición de campos directamente en pantalla.

Para definir los campos directamente en pantalla primero se especifica el carácter delimitador seguido por las posiciones que ocupará el campo. Los siguientes caracteres se usan para dar las posiciones del campo:

Carácter	Significado
9	Posición numérica
0	Posición numérica justificado a la derecha
.	Notación decimal (únicamente para campos numéricos)
S	Posición de signo (únicamente para campos numéricos)
HH	Posición hexadecimal (Campos binarios) (se capturan en grupos de dos)
X	Posición alfanumérica

Se puede especificar un factor de repetición (n), de la forma X(n).
Ejemplo : X(6) es equivalente a XXXXXX.

Los siguientes son ejemplos de definición de campos.

Campo	Comentario
:999	Campo numérico modificable intenso de tres posiciones
(99999	Campo numérico de salida intenso de cinco posiciones
?TEXTO	Texto intenso
&XXXX	Campo alfanumérico modificable visible de cuatro posiciones
+X(7)	Campo alfanumérico de salida visible de siete posiciones

Seleccionando campos de una vista de usuario o de una definición de datos

Para seleccionar un campo de una vista o de una definición de datos, primero se debe especificar en la entrada " Ob: __ ", el tipo de objeto y el nombre del objeto.

Tipo de Objeto

A	Area de parámetros.	
G	Area global de datos.	
H	Rutina de ayuda.	**
L	Area de datos local.	
M	Mapa.	
N	Subprograma.	**
P	Programa.	**
S	Subrutina.	**
V	Vista	

** Estos objetos se pueden utilizar únicamente si tienen definición de datos (DEFINE DATA).

Una vez que ha sido seleccionada una vista de usuario se puede avanzar o regresar usando los comandos de posición (+, -, ++, --, +n, -n) .

Para seleccionar uno o más campos de la vista de usuario se captura el carácter delimitador seguido del número (lado izquierdo de la vista de usuario) o la letra (lado derecho de la vista de usuario) del campo deseado.

Ejemplos:

Campo	Comentario
&3	Selecciona el campo 3 (lado izquierdo) de la vista de usuario y lo pone como modificable.
(C	Selecciona el campo C (lado derecho) de la vista de usuario y lo pone como de salida.

Una vez que se seleccionaron los campos y posterior a un <Enter> los campos se mostrarán en el mapa. Los campos que contengan máscaras se presentarán en el mapa como M , MMMMMMM , MMM , etc.,

Los nombres de los campos seleccionados serán iguales a los nombres de los campos de la vista seleccionada.

Utilizando variables del sistema.

Las variables del sistema (*TIME, *DATE, *USER, *USER-NAME, *LINE-SIZE, etc.) se especifican como sigue; primero se pone el carácter delimitador de salida seguido del nombre de la variable del sistema.

Ejemplos :

(*DATE (*USER-NAME

Las variables del sistema no se pueden modificar ; únicamente se usan como campos de salida .

Ejemplo de diseño de pantallas:

```
Ob: __        Ob:   D    *DEL    CLS    ATTR    *DEL    CLS    ATTR
.             .          BLNK    T      D       ?       T      I
.             .          _       A      D       )       A      I
.             .          +       O      D       (       O      I
.             .          &       M      D       :       M      I
.             .
.             .
001  - 010---+---+---+--030---+---+---+--050--+---+---+---070--- +
 (XXXXXXXX                                                (XXXXXXXX
                      PANTALLA DE EJEMPLO
-----------------------------------------------------------------------
              ?CLAVE DEL ARTICULO: :XXXXXXX
             ?NOMBRE DEL ARTICULO: :XXXXXXXXXX
                     ?DESCRIPCION: :XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
          ?CANTIDAD EN EXISTENCIA: :99999
      ?CLAVES  DE  DISTRIBUIDORES: :XXXXX    :XXXXX    :XXXXX

Enter -PF1-PF2-PF3-PF4-PF5-PF6-PF7-PF8- -PF9- -PF10- -PF11--PF12-
HELP  Mset Quit Test  Edit  top  -  +- Full   <       >     Let
```

Si se realiza un Test (prueba) con la tecla de función <PF4>, la pantalla se desplegará como sigue:

```
MAPART01                                                    MAPART01
                      PANTALLA DE EJEMPLO
-----------------------------------------------------------------------
              CLAVE DEL ARTICULO: _______
             NOMBRE DEL ARTICULO: __________
                     DESCRIPCION: ______________________
          CANTIDAD EN EXISTENCIA: _____
      CLAVES  DE  DISTRIBUIDORES: _____    _____    _____

Enter -PF1-PF2-PF3--PF4--PF5--PF6--PF7--PF8--PF9--PF10--PF11-PF12-
HELP   Mset Quit est  Edit  Top     +- Full  <       >       Let
```

Para agregar los nombres a los campos se invoca la extensión de edición de campos mediante el comando de campo " .E " posicionado en el campo en el que se desea editar.

Además de agregar o modificar el nombre del campo y su longitud, se pueden agregar otras características como: máscaras, rutinas de ayuda, variables de control, etc.

La siguiente pantalla se desplegará cuando este comando se ejecute.

```
Fld P-CVE-ARTICULO                                          Fmt   A7
-----------------------------------------------------------------------
AD=MIT'_' ___    ZP=OFF     SG=OFF    HE=____________      Rls   0
AL= _____        CD=____    CV=____________________        Mod User
PM=___                      DY=____________________________
EM=____________________________________________________________

001    -010--+---+ --+--030--+---+---+--050--+---+---+---070--+---
(XXXXXXXX                                                (XXXXXXXX
                      ?PANTALLA DE EJEMPLO
-----------------------------------------------------------------------
              ?CLAVE DEL ARTICULO: .EXXXXXX
             ?NOMBRE DEL ARTICULO: :XXXXXXXXXX
                     ?DESCRIPCION: :XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
          ?CANTIDAD EN EXISTENCIA: :99999
      ?CLAVES  DE  DISTRIBUIDORES: :XXXXX    :XXXXX    :XXXXX

Enter PF1-PF2- PF3--PF4--PF5--PF6--PF7-PF8-PF9--PF10--PF11--PF12-
HELP  Mset Quit test Edit Top  --     +  Full   <       >   Let
```

Lo siguiente es una descripción de cada entrada de la pantalla de edición de campos.

Entrada	Descripción
Fld	Nombre del campo. Si el campo fue tomado de una vista de usuario, el nombre del campo contendrá como prefijo el nombre de la vista seguido del nombre.
Fmt	Formato y longitud del campo.
AD	Atributos del campo (intenso, modificable, etc.)
ZP	OFF indica que los ceros a la izquierda de los valores no serán impresos. ON indica que los ceros a la izquierda de los valores serán impresos.

060037

SG	OFF indica que la posición del signo no será colocada, ON indica que sí será colocada.
HE	El nombre de una rutina de ayuda que se asigna al campo, se invoca con el signo +. En la ventana auxiliar se declara el nombre de la rutina de ayuda y los parámetros que utiliza.
Rls	Reglas de proceso actualmente definidas para el campo.
AL/NL	Se define la longitud del campo que será desplegada.
CD	Atributos de color.
CV	Variable de control. El nombre de la variable la cual contendrá los atributos a ser usados por este campo.
Mod	Indica como fueron creadas las variables. DATA el campo fue seleccionado de una definición de datos (DEFINE DATA). SYS es una variable del sistema. UNDEF fue creado directamente en la pantalla y tiene un alias como nombre. USER el nombre del campo se creó mediante la extensión de edición de campos. VIEW el campo fue seleccionado de una vista de usuario.
PM	Modo de impresión. C indica que será usado un carácter alterno, I indica dirección de impresión inversa y N se permite desplegar, pero no se permite imprimir.
DY	Atributos de cadena dinámicos. Se definen caracteres contenidos en una cadena de texto de una variable alfanumérica para controlar el conjunto de atributos.
EM	Máscara de edición a ser usada por el campo.

Definición de arreglos

Una definición de arreglo es inicializado por el comando '.A' . Se captura en el campo que se definirá como un arreglo. Al ejecutar este comando aparece la siguiente pantalla.

```
Name ÑP-CVE-DISTRIBUIDOR          Upper  Bnds   3___   1___   1___
-----------------------------------------------------------------
Dimensions              Ocurrences    Starting from          Spacing
0 . Index  vertical      1___          _____          0.     Lines
1 . Index  horizontal    3___          _____          4      Columns
0 . Index  (h/v) V       1___          _____          0     Cls/Ls

001    -010--+---+ --+--030--+---+---+--050--+---+---+--070--+---
(XXXXXXXX                                                (XXXXXXXX
                        PANTALLA DE EJEMPLO
-----------------------------------------------------------------
              ?CLAVE DEL ARTICULO: .EXXXXXX
             ?NOMBRE DEL ARTICULO: :XXXXXXXXXX
                   ?DESCRIPCION: :XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
          ?CANTIDAD EN EXISTENCIA: :99999
      ?CLAVES  DE  DISTRIBUIDORES: .AXXXX    :XXXXX    :XXXXX

Enter -PF1-PF2-PF3-PF4--PF5--PF6--PF7--PF8--PF9--PF10---PF11 PF12-

HELP  Mset Quit Test Edit Top     +   Full   <     >     Let
```

Name	Nombre del campo.
Upper Bnds	Tamaño de las dimensiones del arreglo (únicamente tres dimensiones). Es el tamaño de un arreglo como se define en una vista de usuario o como se define un arreglo de usuario.
Dimensions	Un arreglo puede tener hasta tres dimensiones. El orden en el cual las dimensiones de el arreglo son presentadas por los valores que se ponen a la izquierda de las entradas: índice vertical, índice horizontal e índice fijo.
Ocurrences	El número de ocurrencias que se presentan en la pantalla a ser definidas para una dimensión. No se usa para arreglos de tres dimensiones porque únicamente se despliegan dos dimensiones en la pantalla.
Starting from	El valor de inicio del índice para una dimensión. Se puede utilizar un valor numérico o una variable numérica, si esta variable no se define en el mapa toma el formato numérico (N) y de longitud 7.
Spacing	El número de líneas blancas (dimensión vertical) o columnas blancas (dimensión horizontal) que se insertan entre cada ocurrencia de la dimensión.

4. IMPLEMENTACION DE PANTALLAS

4.1. Invocando una pantalla

Con frecuencia es necesario implementar dentro de un programa, aquellos procesos que se encuentran relacionados con la entrada y salida de información vía pantalla. Tal es el caso de un programa que tiene como función actualizar los registros de un archivo de la base de datos (altas, bajas, cambios y consultas).

Dentro del ambiente de Natural, un diseño de pantalla se llama Mapa y se crea por medio del Editor correspondiente, como se vio en el Capítulo "Editores". Aunque es posible codificar un proceso que requiere entrada/salida de información por pantalla sin el uso de mapas, se recomienda por modularidad utilizar esta facilidad que brinda el modo estructurado del lenguaje Natural.

Para invocar un mapa se utiliza la sentencia INPUT y tiene el siguiente formato:

```
INPUT [WITH TEXT]
[MARK]
[ALARM]
USING nombre de mapa [NO ERASE]
{operando1 [(parámetros)] }
```

La forma más frecuente de utilizar la instrucción INPUT es la siguiente:

```
INPUT USING MAP 'nombre_del_mapa'
```

USING MAP invoca un mapa previamente definido. Un mapa debe ser creado y catalogado previamente a la compilación del programa que lo invoca. Cada vez que el diseño del mapa es modificado a través del editor de mapas, este deberá ser catalogado y los programas que hagan referencia al mapa deberán ser recompilados.

El nombre del mapa puede ser una constante alfanumérica o variable definida por el usuario con una longitud máxima de 8 caracteres. En el caso de una variable, esta debe haber sido definida previamente e inicializada con el nombre del programa.

Si se añade la opción **ALARM** dentro de la instrucción INPUT, se genera la emisión de un sonido de la terminal, en tanto que la opción MARK posiciona el cursor en el nombre del campo como referencia.

Ejemplo:

```
INPUT MARK *CVE-ARTICULO ALARM USING MAP
'ARTIC001'
```

La opción **NO ERASE** despliega un mapa sin borrar el contenido actual de la pantalla. Ejemplo:

```
INPUT USING MAP 'ARTIC001' NO ERASE
```

Cuando se quiere desplegar un texto en la línea de mensajes del mapa, se añade la opción **WITH TEXT**, como se ilustra en el siguiente ejemplo:

```
0010 DEFINE DATA LOCAL
0020 01 #WA-MENSAJE          (A40)
0030 01 #WA-MAP-ART          (A12)
0040 01 REDEFINE             #WA-MAP
0050    02 #WA-CVE-ART       (A07)
0060    02 #WN-CANT          (N05)
0070 END-DEFINE
0080 ************************************************
0090 *          INICIO DE PROGRAMA
0100 ************************************************
0110 MOVE '...TECLA INVALIDA...'  TO  #WA-MENSAJE
0120 INPUT WITH TEXT #WA-MENSAJE USING MAP 'ARTIC001'
0130 ....
0140 ....
0260 ************************************************
0270 *          FIN DE PROGRAMA
0280 ************************************************
0290 END
```

También es posible desplegar un mensaje en el mapa haciendo referencia al número de mensaje dentro de un archivo de mensajes de Natural. Ejemplo:

```
0010  DEFINE DATA LOCAL
0020  01 #WN-NUM-MENSAJE      (N05)
0030  01 #WA-CVE-ART          (A07)
0040  END-DEFINE
0050  ****************************************
0060  **                INICIO DE PROGRAMA
0070  ****************************************
0080  ...
0090  MOVE  45  TO  #WN-NUM-MENSAJE
0100  INPUT WITH TEXT #WN-NUM-MENSAJE
0110     MARK *#WA-CVE-ART USING MAP 'ARTIC001'
0120  ...
0130  ****************************************
0140  **                FIN DE PROGRAMA
0150  ****************************************
0160  END
```

En este ejemplo se invoca al mapa 'ARTIC001', se despliega el mensaje de texto número 45 del archivo de mensajes de NATURAL y coloca el cursor en el campo ÑWA-CVE-ART del mapa.

4.2 Uso de teclas de función (PF's).

En la sección anterior se ha descrito como desplegar un mapa. Ahora bien, continuando con el proceso para implementar una pantalla, se hace necesaria la programación de las teclas de función.

Las teclas de función (Program Function Keys), se usan para asignar funciones específicas que son activadas durante la ejecución de un programa, tales como la ayuda, salida, avanzar o retroceder páginas, etc. El lenguaje Natural cuenta con la instrucción SET KEY para este fin.

Cuando la instrucción SET KEY es procesada, NATURAL recibe el control de las teclas durante la ejecución del programa y usa los valores asignados a las teclas. Las teclas de función pueden ser asignadas de tal forma que:

- Sean sensitivas al programa,
- Contengan nombres de programa,
- Ejecuten comandos de la terminal.

A continuación se listan los formatos para la instrucción SET KEY:

```
1. SET KEY  { ON
              OFF
              ALL }

2. SET KEY { { PAn   [ { PGM                    ]  [ NAMED { OFF      } ] }
               PFn       ON                                  operand1
               CLR }     = OFF
                         [ DATA ] operand1
                         HELP }
```

SET KEY OFF	Desactiva todas las asignaciones hechas a las teclas de función y regresa el control de las teclas sistema monitor TP.
SET KEY ON	Reactiva el comando o nombre de programa asignado a las teclas que tuvieron una asignación y fueron entonces hechas sensitivas al programa.
SET KEY ALL	Provoca que todas las teclas sean hechas sensitivas al programa.
SET KEY PA2 = HELP	Provoca que la facilidad HELP sea invocada para un campo en el cual esté posicionado el cursor cuando <PA2> es presionada.
SET KEY PF8 = OFF	Desactiva <PF8> y regresa el control de la tecla de función al monitor TP.
SET KEY PF9 = ON	Reactiva el comando o programa asignado a <PF9>. Si ningún valor ha sido asignado a la tecla, esto será equivalente a pulsar <ENTER>.
SET KEY PF12 = DATA 'string'	La cadena de datos especificada es asignada a <PF12>. Cuando <PF12> es usada, la cadena de datos es colocada dentro del campo de entrada (INPUT) en el que el cursor esté posicionado actualmente.

Teclas sensitivas al programa

Si una tecla <PF> o <PA> es hecha sensitiva al programa, por ejemplo SET KEY PF6 = PGM, pulsar <PF6> tiene el mismo efecto que pulsar <ENTER>. Todos los datos que hayan sido entrados sobre la pantalla son transferidos al programa. La variable del sistema *PF-KEY identifica cuál tecla fue presionada.

Asignando nombres de programas o comandos de la terminal.

Si la asignación de una tecla permanece en efecto para más de un programa, un comando puede ser asignado a la tecla. La programación de la tecla permanecerá en efecto cuando el estado de ejecución del programa sea dejado y entonces regrese el control. Cuando la tecla es pulsada, cualquier programa en ejecución es interrumpido y el comando asignado a la tecla es invocado. De igual forma, un nombre de programa puede ser asignado a la tecla de función.

Ejemplo:

```
0010 DEFINE DATA LOCAL
0020 01 ÑWA-PROGRAMA          (A08)
0030 END-DEFINE
0040 SET KEY PF1 = 'AYUDA-P1' NAMED 'AYUDA'
0050         PF2 = 'LISTA-P1' NAMED 'LISTA'
0060         PF3 = 'SALE-P1'  NAMED 'FIN'
0070         PF5 = 'CONT-P1'  NAMED 'CONT'
0080         PF6 = 'REGR-P1'  NAMED 'REGR'.
0090 ******************************************
0100 **        INICIO DE PROGRAMA
0110 ******************************************
0120 MOVE *PROGRAM TO ÑWA-PROGRAMA
0130 INPUT USING MAP 'ARTIC001'
0190 ...
0200 ******************************************
0210 **        FIN DE PROGRAMA
0220 ******************************************
0230 END
```

4.3 Técnicas de validación.

La validación de los campos de un mapa es necesaria para verificar que la información proporcionada en cada campo de entrada de dicho mapa corresponda en formato y contenido con los requerimientos de análisis y diseño, como es el caso de su correspondencia con la información existente en archivos del sistema. Por ejemplo, puede validarse un campo con las características de no vacío, numérico y dentro de un rango de valores. El proceso de validación puede hacerse de diferentes formas, aunque básicamente serán una combinación de sentencias DECIDE, IF-THEN-ELSE, REINPUT y MARK, como se muestra a continuación:

```
0010   DEFINE DATA LOCAL
0020   01 ÑWA-MAPA            (A22)
0030   01 REDEFINE ÑWA-MAPA
0040      02 ÑWA-CVE-ART      (A07)
0050      02 ÑWA-NOMBRE-ART   (A10)
0060      02 ÑWN-CANT         (N05)
0070   ...
0080   END-DEFINE
0090   ********************************************
0100   **            INICIO DE PROGRAMA
0110   ********************************************
0120   ...
0130   REPEAT
0140         INPUT USING MAP  'ARTIC001'
0150         DECIDE ON FIRST VALUE OF *PF-KEY
0160                VALUE 'ENTR'
0170                      IGNORE
0180                VALUE 'PF3'
0190                      ESCAPE BOTTON
0200                NONE
0210                      REINPUT ' TECLA NO VALIDA ' ALARM
0220         END-DECIDE
0230         IF ÑWA-CVE-ART = ' ' THEN
0240               REINPUT 'CLAVE DE ARTICULO REQUERIDA'
0250               MARK *ÑWA-CVE-ART ALARM
0260         END-IF
0270         FIND VIEW-ARTICULOS WITH CVE-ART EQ ÑWA-CVE-ART
0280               IF NO RECORDS FOUND
0290                     REINPUT 'NO EXISTE CVE.ART.'
0300                     MARK *ÑWA-CVE-ART
0310               END-NOREC
0320         END-FIND
0330         IF ÑWA-NOMBRE-ART = ' ' THEN
0340               REINPUT 'NOMBRE DEL ARTICULO REQUERIDO'
0350               MARK *ÑWA-NOMBRE-ART
0360         END-IF
0370         IF ÑWN-CANT NE MASK(NNNNN) THEN
0380               REINPUT 'CANTIDAD DEBE SER NUMERICA'
0390               MARK *ÑWN-CANT
0400         END-IF
0410         PERFORM PROCESAR-INFORMACION
0420   END-REPEAT
0430   *
0440   ********************************************
0450   **            FIN DE PROGRAMA
0460   ********************************************
0470   END
```

En este ejemplo, la cláusula REPEAT es empleada para ejecutar n veces el llamado del mapa y sus respectivas validaciones hasta que el usuario presiona la tecla de función <PF3>.

La instrucción DECIDE evalúa las teclas de función, que para el presente ejemplo sólo son válidas <PF3> y <ENTER>.

000041

La secuencia de IF's marca propiamente la estructura de validación, que varía en cada programa dependiendo de la definición de variables en el mapa y los valores que deban tener al ser capturadas en el mapa.

Para verificar que la información capturada exista en la base de datos, se realizan búsquedas a los archivos; para este ejemplo se utilizó la instrucción FIND.

Dentro de cada sentencia IF mostrada en el ejemplo, encontramos las cláusulas REINPUT y MARK, las cuales serán explicadas con detalle a continuación.

REINPUT

La instrucción REINPUT regresa la ejecución a la instrucción INPUT previamente ejecutada. Es usada generalmente para desplegar un mensaje indicando que el dato de entrada como resultado de la instrucción INPUT previa fué inválido.

Las instrucciones WRITE o DISPLAY no deberán ser ejecutadas entre la sentencia INPUT y sus correspondientes REINPUT's. La instrucción REINPUT no es válida en modo batch.

Después de ejecutar la instrucción REINPUT, el control del programa retoma a la subrutina, condición especial o al ciclo en proceso que existía cuando la instrucción INPUT fué ejecutada. Si el ciclo inició después de la ejecución del INPUT y la sentencia REINPUT se encuentra dentro de éste, el ciclo será interrumpido y reestablecido después de que la instrucción INPUT haya sido reprocesada como resultado del REINPUT.

Si una jerarquía de subrutinas fue invocada después de la ejecución de la instrucción INPUT, y el REINPUT es ejecutado dentro de una subrutina, NATURAL buscará de regreso todas las subrutinas automáticamente y reposicionará el programa en la instrucción INPUT.

No es posible, sin embargo, tener una instrucción INPUT posicionada dentro de un ciclo, una subrutina o bloque especial de condiciones, y entonces procesar la instrucción REINPUT cuando el estado bajo el cual la instrucción INPUT fué ejecutada, ha sido terminada. Entonces se producirá un mensaje y el programa terminará cuando esta condición de error sea detectada.

El formato de la instrucción REINPUT es el siguiente:

```
REINPUT [{ parámetros de la instrucción }]
        [      USING HELP              ]
        [      [WITH TEXT]             ]
               [ MARK ]
               [ ALARM ]
```

Parámetros de la instrucción

Los parámetros especificados en una instrucción REINPUT serán aplicados a todos los campos de la pantalla producidos por la última instrucción INPUT.

Cualquier parámetro especificado en el nivel de campo, (ver la opción MARK), sobrescribirá cualquier parámetro correspondiente en el nivel de la instrucción.

USING HELP

Causa que la rutina de HELP sea invocada. Ejemplo:

REINPUT USING HELP MARK 5

Resultado: La subrutina de ayuda para el quinto campo es invocada.

Las opciones WITH TEXT, MARK y ALARM, tienen un uso equivalente a las descritas en la instrucción INPUT en la primera parte del presente capítulo.

Ejemplo:

```
0010 *  ASIGNACION DE ATRIBUTOS EN LA INSTRUCCION REINPUT
0020 DEFINE DATA LOCAL
0030 01 #WA-NOMBRE-ART        (A10)
0040 01 #WN-CANT              (N05)
0050 END-DEFINE
0060 *
0070 ***************************************
0080 **        INICIO DE PROGRAMA
0090 ***************************************
0100 INPUT (AD=AU)
0110          'NOMBRE DEL ARTICULO: ' #WA-NOMBRE-ART
0120          'CANTIDAD : '  #WN-CANT
0130 IF #WA-NOMBRE-ART = ' ' OR #WN-CANT = 0 THEN
0140    REINPUT 'DEBERA CAPTURAR NOMBRE DEL ARTICULO Y CANTIDAD'
0150      MARK #WA-NOMBRE-ART (AD=A)
0160 END-IF
0170 ...
0180 ***************************************
0190 **        FIN DE PROGRAMA
0200 ***************************************
0210 END
```

Donde :

AD	= Atribute Definition
I	= Intensified
U	= Underlined
P	= Temporary Protect
A	= Input Field

4.4 Técnicas de paginación.

Paginación si nos referimos a sistemas operativos es una técnica de direccionamiento de memoria de bajo nivel.

En programación de sistemas de información, paginación significa desplegar un documento o una lista de información pantalla por pantalla, es decir, solo cierto número de líneas se pueden visualizar a la vez (una página); por lo que debe existir alguna manera de poder ver las siguientes N líneas de información o bien las anteriores N líneas. Este proceso puede consumir recursos considerables de memoria RAM si pretendemos cargar toda nuestra información antes de comenzar a visualizarla. Puede ocurrir que el espacio disponible en memoria no sea suficiente para recibir la cantidad de datos que deseamos consultar y en el peor de los casos un programa suspendería su ejecución enviando un mensaje de error. Por lo tanto, ésta manera de enfocar el problema es ineficiente.

Contamos con dos alternativas de solución, la primera de ellas se lista a continuación.

La idea básica de este algoritmo es realizar completamente la consulta deseada con el fin de obtener los descriptores del primer registro de cada página para guardarlos en una arreglo.

De esta manera podemos en cualquier momento accesar los siguientes N registros o los N anteriores a partir de una llave guardada en dicho índice, ya que se trata de un acceso secuencial por descriptor.

```
0010 ******************
0020 DEFINE DATA
0030 ******************
0040 LOCAL
0050 01 ART VIEW OF ARTICULOS-COMESTIBLES
0060    02 CVE-ART            (A7)
0070    02 NOMBRE-ART         (A10)
0080    02 CVE-PRESNT         (A05)
0090    02 DESCRIP            (A30)
0100    02 CANT-PROD          (N5)
0110 01 ÑWA-CVE                       (A7/1:12)
0120 01 ÑWA-PRESNT             (A5/1:12)
0130 01 ÑWA-DESC                      (A30/1:12)
0140 01 ÑWN-CANT                (N5/1:12)
0150 01 ÑWN-NOMBRE-ART          (A10/1:12)
0160 01 ÑWN-LINEAS-MAPA        (N2)
0170 01 ÑWN-CAMPO              (N1)
0180 01 ÑWN-CONT               (N6)
0190 01 ÑWN-REG-LEI            (N6)
0200 01 ÑWN-PAG                (N4)
0210 01 ÑWN-TOT-PAG            (N4)
0220 01 ÑWA-LLAVES             (A7/1:1000)
0230 01 ÑWA-CVE-I              (A7)
0240 01 ÑWA-CVE-F              (A7)
0250 01 ÑWA-LLA-ANT-I          (A7)
0260 01 ÑWA-LLA-ANT-F          (A7)
0270 01 ÑWA-MSG                (A50)
0280 01 ÑWN-IND-P              (N1)
0290 01 ÑWN-EDO-PRG            (N1)
0300 01 ÑWL-MAS-10000          (L)
0310 END-DEFINE
0320 ************************************
0330 **        INICIO DE PROGRAMA
0340 ************************************
0350 *
0360 /**DECLARACION DE PF'S
0370 *
0380 SET KEY ALL
0390 SET KEY PF2 NAMED 'MenuP'
0400 SET KEY PF3 NAMED 'Anter'
0410 SET KEY PF4 NAMED 'Alta'
0420 SET KEY PF5 NAMED 'Baja'
0430 SET KEY PF6 NAMED 'Modif'
0440 SET KEY PF7 NAMED 'Atras'
0450 SET KEY PF8 NAMED 'Avanz'
0460 SET KEY PF10 NAMED 'Limpia'
0470 PERFORM INICIALIZAR-CAMPOS
0480 *
0490 *
0500 ********************************
0510 /* CUERPO PRINCIPAL DEL PROGRAMA
0520 /********************************
0530 REPEAT
0540     INPUT  WITH  TEXT  *ÑWA-MSG  MARK  ÑWN-CAMPO  USING  MAP
         'MAPART05'
0550     RESET ÑWA-MSG
0560     DECIDE ON FIRST VALUE OF ÑWN-EDO-PRG
0570         VALUE 1
0580                 PERFORM VALIDAR-PFS-INICIO
0590                 PERFORM ACCESAR-ENTIDAD
0600         VALUE 2
0610                 PERFORM VALIDAR-PFS-CONTINUA
0620         NONE
0630                 IGNORE
0640     END-DECIDE
0650     PERFORM ARMAR-MAPA
0660 END-REPEAT
0670 ************************************
```

```
1770 ************************************************
1780 DEFINE SUBROUTINE ARMAR-MAPA
1790 ************************************************
1800 #WN-IND-P := 0
1810 READ (#WN-LINEAS-MAPA) ART BY CVE-ART
1820     STARTING FROM #WA-LLAVES(#WN-PAG)
1830        ADD 1 TO #WN-IND-P
1840        MOVE CVE-ART    TO #WA-CVE (#WN-IND-P)
1850        MOVE PRESNT     TO #WA-PRESNT (#WN-IND-P)
1860        MOVE DESCRIP    TO #WA-DESC (#WN-IND-P)
1870        MOVE CANT       TO #WN-CANT (#WN-IND-P)
1880        MOVE NOMBRE-ART TO #WN-NOMBRE-ART (#WN-IND-P)
1890 END-READ
1900 END-SUBROUTINE
1910 ************************************************
1920 **        FIN DE PROGRAMA
1930 ************************************************
1940 END
```

Como ventajas podemos mencionar las siguientes:

- Permite accesar 1000 páginas de información aproximadamente en una sola consulta. Pueden ser más o menos dependiendo del tamaño del descriptor y del número de líneas por página (en este ejemplo fueron 10000 registros). Por ejemplo si la llave tuviera una longitud de 20 caracteres alfanuméricos con 10 líneas por página, serían como máximo 600 páginas (el tamaño del arreglo es limitado), es decir, 6000 registros. Si aumentamos el número de líneas por página a 15 obtendríamos las mismas 600 páginas pero accesaríamos a 9000 registros.

- Se envía el mapa una sola vez dentro de la lógica del programa.

- Evita el uso innecesario de recursos utilizando la instrucción HISTOGRAM para obtener las llaves.

El siguiente programa también realiza la consulta por completo como primer paso, pero este obtiene los ISN's de todos los registros leídos, lo que permite a su vez efectuar la paginación. Puede observarse el uso de variables de control con el fin de permitir la captura de la llave al inicio de la consulta, e impedir capturas no deseadas en el área donde los datos serán desplegados; se protege el campo llave para que no pueda ser modificado mientras la información esté siendo desplegada y se esté efectuando la paginación.

```
0010 ******************
0020 DEFINE DATA
0030 ******************
0040 LOCAL
0050 * ---------------- DEFINICION DE VISTAS ----------------*
0060 01  VIEW-ARTICULOS VIEW OF ARTICULOS-COMESTIBLES
0070     02 CVE-ART
0080     02 NOMBRE-ART
0090     02 CVE-PRESNT
0100 * ---------------- VARIABLES DEL MAPA ----------------*
0110 01 #WA-MAPA
0120 01 REDEFINE                      #WA-MAPA
0130      02 #WM-CVE-ART              (A7/1:12)
0140      02 ...
0150      02 ...
0160 * ----------DEFINICION DE VARIABLES AUXILIARES ----------*
0170 01 #WP-ISN                       (P10/300)
0180 01 #WN-MAX-ISN          (N04)
0190 01 #WN-CONT-ISN         (N04)
0200 01 #WN-TOPE-ISN         (N04)  INIT <300>
0210 01 #WN-TOPE-PAG         (N04)  INIT <30>
0220 01 #WN-TAM-PAG          (N02)  INIT <10>
0230 01 #WN-NUM-PAG          (N03)
0240 01 #WN-MAX-PAG          (N03)
0250 01 #WN-CONT-MAPA        (N02)
0260 01 #WC-LLAVE            (D)
0270 01 #WC-DATOS            (D)
0280 END-DEFINE
0290 ************************************************
0300 **      . INICIO DE PROGRAMA
0310 ************************************************
0320 REPEAT
0330      MOVE  (AD=ID)  TO  #WC-LLAVE
0340      MOVE  (AD=P)   TO  #WC-DATOS
0350      INPUT USING MAP 'MARTIC02'
0360      DECIDE ON FIRST VALUE OF *PF-KEY
0370             VALUE 'PF3' ESCAPE BOTTOM
0380             VALUE 'PF12'        TERMINATE
0390             VALUE 'ENTR'        PERFORM  LIMPIA-MAPA
0400                                 PERFORM  CONSULTA
0410             NONE IGNORE
0420      END-DECIDE
0430      MOVE  (AD=P)    TO  #WC-LLAVE
0440      MOVE  (AD=ID)   TO  #WC-DATOS
0450      REPEAT
0460           INPUT USING MAP 'MARTIC02'
0470           DECIDE ON FIRST VALUE OF *PF-KEY
0480                VALUE  'PF3'   ESCAPE BOTTOM
0490                VALUE  'PF12'  TERMINATE
0500                VALUE  'PF7'   PERFORM  PAG-AT
0510                VALUE  'PF8'   PERFORM  PAG-AD
0520                VALUE  'ENTR'  PERFORM  PROCESA
0530                NONE   IGNORE
0540           END-DECIDE
0550      END-REPEAT
```

```
0560 END-REPEAT
0570 ******************************
0580 DEFINE SUBROUTINE CONSULTA
0590 ******************************
0600 READ VIEW-ARTICULOS BY CVE-ART EQ = ÑWM-CVE-ART(ÑWN-CONT-
     MAPA)
0610            ADD 1 TO ÑWN-CONT-ISN
0620            IF ÑWN-CONT-ISN GT ÑWN-TOPE-ISN
0630                 SUBTRACT 1 FROM ÑWN-CONT-ISN
0640                 ESCAPE BOTTOM
0650            END-IF
0660            MOVE *ISN TO ÑWP-ISN (ÑWN-CONT-ISN)
0670 END-READ
0680 IF ÑWN-CONT-ISN = 0
0690       REINPUT 'NO EXISTEN REGISTROS '
0700 END-IF
0710 MOVE ÑWN-CONT-ISN TO ÑWN-MAX-ISN
0720 COMPUTE ÑWN-MAX-PAG = ((ÑWN-MAX-ISN - 1)/ÑWN-TAM-PAG) + 1
0730 ASSIGN ÑWN-NUM-PAG = 1
0740 ASSIGN ÑWN-CONT-ISN = 0
0750 PERFORM ARMA-MAPA
0760 END-SUBROUTINE
0770 ******************************
0780 DEFINE SUBROUTINE ARMA-MAPA
0790 ******************************
0800 PERFORM LIMPIA-MAPA
0810 FOR ÑWN-CONT-MAPA = 1 TO ÑWN-TAM-PAG
0820      ADD 1 TO ÑWN-CONT-ISN
0830      IF ÑWN-CONT-ISN GT ÑWN-MAX-ISN
0840           ESCAPE BOTTOM
0850      END-IF
0860      GET VIEW-ARTICULOS ÑWP-ISN(ÑWN-CONT-ISN)
0870      MOVE CVE-ART TO ÑWM-CVE-ART(ÑWN-CONT-MAPA)
0880      MOVE ........TO ..........(ÑWN-CONT-MAPA)
0890      .......
0900      .......
0910 END-FOR
0920 END-SUBROUTINE
0930 ******************************
0940 DEFINE SUBROUTINE PAG-AT
0950 ******************************
0960 IF ÑWN-NUM-PAG LE 1
0970    REINPUT 'NO HAY PAGINA ANTERIOR'
0980 END-IF
0990 SUBTRACT 1 FROM ÑWN-NUM-PAG
1000 COMPUTE ÑWN-CONT-ISN = (ÑWN-NUM-PAG - 1) * ÑWN-TAM-PAG
1010 PERFORM ARMA-MAPA
1020 END-SUBROUTINE
1030 ******************************
1040 DEFINE SUBROUTINE PAG-AD
1050 ******************************
1060 IF ÑWN-NUM-PAG GE ÑWN-MAX-PAG
1070    REINPUT 'NO HAY MAS PAGINAS'
1080 END-IF
1090 ADD 1 TO ÑWN-NUM-PAG
1100 COMPUTE ÑWN-CONT-ISN = (ÑWN-NUM-PAG - 1) * ÑWN-TAM-PAG
1110 PERFORM ARMA-MAPA
1120 END-SUBROUTINE
1130 ************************
1140 DEFINE LIMPIA-MAPA
1150 ************************
1160 RESET CAMPO-MAPA (*)
1170 END-SUBROUTINE
1180 ******************************
1190 DEFINE SUBROUTINE PROCESA
1200 ******************************
1210 ...
1220 ...
1230 ...
1240 END-SUBROUTINE
1250 ****************************************
1260 **        FIN DE PROGRAMA
1270 ****************************************
1280 END
```

Existen algunas limitantes importantes para estos dos algoritmos:

- Si alguien borra algún registro o varios el primer programa podría desplegar información repetida entre una página y la precedente
- El segundo programa definitivamente abortará cuando se ejecute un GET sin que el registro exista.
- Si desde otra terminal se dan de alta más registros, ninguno de los dos programas presentará los cambios efectuados.
- El tamaño de un arreglo está bastante limitado en la mayoría de los centros de cómputo (12,000 bytes).
- En el caso del primer algoritmo si la llave no es única; la paginación no tendrá el efecto deseado.NN

5. MODULARIDAD

Modularidad es una técnica de diseño que consiste en dividir las funciones de un sistema en partes con la finalidad de hacer más comprensible el requerimiento solicitado por el usuario, tanto para su desarrollo (análisis, diseño, programación, pruebas e implementación), como para su mantenimiento.

Dicho en otras palabras, la modularidad consiste en separar una función en módulos, si la función que realiza cada uno de ellos es muy compleja se debe efectuar otra división, a partir de los ya existentes, hasta que la operación de cada submódulo sea lo más sencilla y clara.

Para la modularidad enfocada a la programación, se deben considerar las siguientes cinco características:

- Descomposición modular. Es un método de diseño el cual consiste en separar el problema en varios subproblemas, cuya solución pueda realizarse por separado, de manera clara y entendible.
- Integridad. Consiste en que un cambio en la especificación de un módulo no impacte en la función de otros módulos.
- Módulos Consistentes. Los módulos deben tener la misma estructura, los mismos estándares.
- Comunicación entre Módulos. La comunicación entre módulos se hace por medio de interfases. Cada módulo debe tener el menor número de interfases posibles.
- Información oculta. Toda la información relativa a un módulo debe ser privativa excepto para algunas interfases.

Para representar gráficamente los módulos o funciones que tiene un programa se utiliza la técnica de Arbol de Módulos, que consiste en diagramar en forma esquemática la organización y jerarquía que presentan ésos de manera lógica dentro de una aplicación (tienen el mismo formato que un organigrama). Los módulos se representan por rectángulos y se conectan a otros de los cuales dependen exclusiva y directamente.

La programación estructurada es una técnica en la cual se aplica la modularidad al identificar el problema central y dividirlo en subproblemas. Una técnica de programación estructurada es la metodología top-down (desarrollo descendente o de arriba-abajo). Dicha técnica nos es útil para la comprensión del requerimiento y para la codificación de la solución, resultando fácil de identificar las principales rutinas y poder hacer un seguimiento claro de su lógica.

Como se mencionó en el Capítulo I, el lenguaje Natural tiene dos formas de desarrollo de programas: modo estructurado y modo reporte. El modo estructurado es el que se recomienda utilizar y es necesario definir dos secciones básicas como se muestra a continuación :

```
DEFINE DATA
...                    (Area de Definición de Datos)
END-DEFINE
...                    (Area de Sentencias.Natural)
END
```

- **Area de Definición de Datos**

Contiene la definición de vistas (archivos de la base de datos), variables auxiliares (campos numéricos, alfanuméricos, empacados, etc.), áreas locales , áreas globales y de parámetros.

- **Area de Sentencias Natural**

Son todas las instrucciones escritas en el lenguaje Natural (FIND, IF, MOVE, FOR, WRITE, etc.).

El desarrollo de programas en modo estructurado, implica un mayor orden y claridad en la escritura de los programas, a diferencia del modo reporte. El modo reporte no obliga al programador a definir área de datos y las variables pueden ser definidas en cualquier parte del programa, lo que dificulta el seguimiento de errores en caso de mantenimiento, razón por la cual se recomienda el modo estructurado.

Los objetos del lenguaje Natural que nos ayudan a la escritura de programas bajo el concepto de modularidad son :

- Subprogramas
- Subrutinas
- Mapas
- Area Local de Datos
- Area Global de Datos
- Rutinas de Ayuda (Helproutine)
- Códigos de Copia (Copycode)

A continuación se describirá el uso de cada uno de ellos dentro del lenguaje Natural.

5. 1 Subprograma

En ciertos casos, al analizar las diferentes funciones del programa que estamos escribiendo, nos percatamos que tenemos uno o varios procesos que son extensos y consecuentemente incrementan la cantidad de líneas de tal forma que resulta muy difícil hacer un seguimiento del mismo. Para ello se recomienda el uso de subprogramas.

Un subprograma es un conjunto de instrucciones que es ejecutado desde uno o varios programas. En Natural, un subprograma tiene la misma estructura que un programa (Area de Definición de Datos y Area de Sentencias Natural). El comando para Crear/Editar un subprograma ha sido previamente descrito en el capítulo "Editores".

Un subprograma se invoca en lenguaje Natural por medio de la instrucción 'CALLNAT'. El paso de parámetros entre el subprograma y el objeto que lo invoca (programa, subprograma, subrutina o copycode), se realiza a través de un Area de Parámetros, la cual puede ser definida dentro del subprograma o en un Area de Parámetros Externa.

A continuación se muestra el formato de la instrucción CALLNAT:

CALLNAT nombre_subprograma. { parámetros [(AD= { M / O })] }

Nombre del subprograma:

Nombre constante (o variable) del subprograma a ser llamado.

Parámetros:

El subprograma Natural debe comenzar con la sentencia DEFINE DATA PARAMETER, la cual define la lista de parámetros que el subprograma recibirá.

La secuencia, formato y longitud de los parámetros deberá coincidir exactamente con la lista de parámetros del programa que hace el llamado al subprograma.

Los parámetros pueden definirse como de sólo lectura (AD=O) o modificables (AD=M). El default es (AD=M).

Observaciones:

- Un subprograma debe ser invocado únicamente vía CALLNAT. No puede ejecutarse como programa principal.
- Un subprograma puede contener una o más sentencias CALLNAT para propósito de llamar a otros subprogramas.
- Un área de datos global la cual está disponible en el programa que invoca NO está disponible en el subprograma.

Ejemplo de llamado a un subprograma:

```
0010 ************************************************************
0020 *                EJEMPLO DE USO DE SUBPROGRAMAS
0030 *FUNCION: LECTURA DE UN ARCHIVO SECUENCIAL Y LLAMADO A UN
0040 *         SUBPROGRAMA PARA PROCESAR LA INFORMACION LEIDA
0050 *PROGRAMA:       SISCOM09
0060 ************************************************************
0070 DEFINE DATA
0080 LOCAL
0090 * ------------------ ARCHIVO DE ENTRADA --------------------*
0100 01 #WA-REG-ARTICULOS            (A12)
0110 01 REDEFINE       #WA-REG-ARTICULOS
0120    02 #WA-CVE-ART        (A07)
0130    02 #WN-CANT           (N05)
0140 * ---------- DEFINICION DE VARIABLES AUXILIARES ----------*
0150 01 #WA-FECHA                   (A08)
0160 01 #WN-TOT-REG-LEIDOS          (N06)
0170 END-DEFINE
0180 ****************************************
0190 **        INICIO DE PROGRAMA          **
0200 ****************************************
0210 MOVE *DATN TO #WA-FECHA  /* FORMATO FECHA (YYYYMMDD)
0220 READ WORK FILE 1 #WA-REG-ARTICULOS
0230          ADD 1 TO #WN-TOT-REG-LEIDOS
0240          CALLNAT SISCOM10 #WA-REG-ARTICULOS #WN-CANT
0250 END-READ
0260 DISPLAY '**** TOTAL DE REGISTROS LEIDOS :  ' #WN-TOT-REG-
LEIDOS
0270 *
0280 ****************************************
0290 **        FIN DE PROGRAMA             **
0300 ****************************************
0310 END
```

En este ejemplo el subprograma invocado se llama 'SISCOM91', el cual recibe como parámetros las variables ÑWA-CVE-ART y ÑWN-CANT, las cuales deben definirse en el Area de Parámetros del Subprograma.

Ejemplo:

```
0010 ************************************************************
0020 *                      SUBPROGRAMA  SISCOM91
0030 *FUNCION: RECIBE LOS PARAMETROS ÑWA-CVE-ART Y ÑWN-CANT,
0040 *         PROCESA EL ARCH. DE LA BASE DE DATOS DE ARTICULOS
0050 *         COMESTIBLES SI LA CANTIDAD SOLICITADA ES MENOR
0060 *         O IGUAL A LA CANTIDAD EN EXISTENCIA
0070 ************************************************************
0080 DEFINE DATA
0090 PARAMETER
0100 * ----------------- DEFINICION DE PARAMETROS ---------------*
0110 01 ÑWA-CVE-ART          (A07)
0120 01 ÑWN-CANT                    (N05)
0130 LOCAL
0140 * ------------------- DEFINICION DE VISTA ------------------*
0150 01 VIEW-ART-COM  VIEW OF ARTICULOS-COMESTIBLES
0160    02 CVE-ART            (A07)
0170    02 CANT                    (N05)
0180 END-DEFINE
0190 ****************************************
0200 **        INICIO DE SUBPROGRAMA **
0210 ****************************************
0220 FIND PARTES WITH CVE-ART EQ  ÑWA-CVE-ART
0230           ACCEPT  IF  ÑWN-CANT LE CANT
...            .....
...            .....
0290 END-FIND
....
....
0320 ****************************************
0330 **        FIN DE SUBPROGRAMA          **
0340 ****************************************
0350 END
```

5.2 Subrutinas

Cuando tenemos un segmento de código que se repite en diferentes partes de un programa, es conveniente crear una subrutina.

Una subrutina es un conjunto de instrucciones Natural que ejecutan una tarea en específico y que se crea con el objetivo de evitar redundancia. Sin embargo, una subrutina puede definirse únicamente con la finalidad de mantener la modularidad del programa, aun cuando no haya redundancia, lo cual facilitará la programación y seguimiento de errores, especialmente cuando el programa es extenso.

Cuando una subrutina es definida dentro de un programa recibe el nombre de Subrutina Interna.

En otras ocasiones, un mismo segmento de código es empleado por diferentes objetos. En tal caso, la subrutina se define fuera del programa y recibe el nombre de Subrutina Externa.

Una subrutina externa es un objeto independiente al programa o programas que la invocan. Las variables utilizadas dentro de la subrutina externa deberán estar definidas como variables locales o globales dentro del programa que hace el llamado.

La forma de Crear/Editar una subrutina externa dentro del ambiente de Natural es descrita en el capítulo "Editores".

Dentro de un programa, la instrucción para llamar una subrutina (interna o externa), es la siguiente:

PERFORM nombre_de_la_subrutina

Nombre de la subrutina:

Debe crearse de acuerdo a las mismas convenciones que un nombre de variable (máximo 32 caracteres).

La subrutina deberá ser definida mediante la sentencia:

DEFINE SUBROUTINE nombre_de_la_subrutina
...
...
...
END-SUBROUTINE

Observaciones:

- Una subrutina puede invocarse a sí misma (recursividad).
- Una subrutina interna tiene acceso a las Areas de Datos Global y Local. Una subrutina externa tiene acceso únicamente al Area Global de Datos.
- Una subrutina puede contener llamadas a otras subrutinas (instrucción PERFORM dentro de la subrutina).

Ejemplo:

```
0010 ************************************************************
0020 *                EJEMPLO DE SUBRUTINA
0030 *FUNCION: LEE EL ARCHIVO 'PRECIOS' Y ACTUALIZA EL DE
0040 *         'ARTICULOS COMESTIBLES'
0050 ************************************************************
0060 DEFINE DATA  LOCAL
0070 * ------------------ DEFINICION DE VISTAS -------------------*
0080 01 VIEW-ART-COM VIEW OF ARTICULOS-COMESTIBLES
0090       02 CVE-ART
0100       02 PRECIO-PUBLICO
0110       02 PRECIO-DISTRIBUIDOR
0120 01 PRECIOS VIEW OF PRECIOS-ARTICULOS
0130       02 CLAVE-ART
0140       02 PREC-PUB
0150       02 PREC-DIST
0160 END-DEFINE
0170 ****************************************
0180 **         INICIO DE PROGRAMA     **
0190 ****************************************
0200 READ PRECIOS BY CLAVE-ART
0210     IF NO RECORDS FOUND
0220       WRITE '*** NO SE ENCONTRARON REGISTROS EN ARCHIVO
           PRECIOS'
0230       ESCAPE-BOTTOM(0200)
0240     END-NOREC
0250     PERFORM ACTUALIZA-ARTICULO
0260 END-READ
0270 *
0280 ************************************
0290 DEFINE SUBROUTINE ACTUALIZA-ARTICULO
0300 ************************************
0310 FIND VIEW-ART-COM WITH CVE-ART EQ CLAVE-ART
0320       IF NO RECORDS FOUND
0330           WRITE '***NO SE ENCONTRARON REG. EN ARCHIVO
               ARTICULOS'
0340           ESCAPE-BOTTOM(0310)
0350       END-NOREC
0360       MOVE PREC-PUB  TO PRECIO-PUBLICO
0370       MOVE PREC-DIST TO PRECIO-DISTRIBUIDOR
0380       UPDATE(0310)
0390       END TRANSACTION
0400 END-FIND
0410 END-SUBROUTINE           /* ACTUALIZA-ARTICULO
0420 ****************************************
0430 **         FIN DE PROGRAMA              **
0440 ****************************************
0450 END
```

5.3 Mapas

En el lenguaje Natural, una pantalla por medio de la cual el usuario interactúa con el sistema para consulta, captura o mantenimiento de la información, recibe el nombre de Mapa.

Los mapas son diseños de pantallas en los cuales se distribuyen los campos a utilizar y se asignan las características a estos campos, por ejemplo: subrayado, intenso, centrado, sólo de entrada, de entrada/salida, sólo de salida; así como el tipo: numérico, alfanumérico, arreglos, etc.

Además de los mapas, una pantalla puede ser definida completamente dentro del programa, sin embargo, se recomienda el uso de los mapas con fines de mantener la modularidad en un programa. Existe una utilería especial dentro de Natural que nos permite la creación y mantenimiento de los mapas: el Editor de Mapas.

La forma de Crear/Editar un mapa es descrita en el capítulo "Editores", donde también se explica con detalle el funcionamiento del Editor de Mapas.

La sentencia INPUT nos permite referenciar un mapa dentro de un programa. Esta sentencia ha sido explicada en el capítulo "Implementación de Pantallas".

A continuación se presenta un ejemplo de referencia a un mapa dentro de un programa :

```
0010 ************************************************************
0020 *                EJEMPLO DE UTILIZACION DE MAPA
0030 *FUNCION: PRESENTAR PANTALLA PARA CAPTURA DE INFORMACION DE
0040 *         ARTICULOS. SE VALIDA LA INFORMACION DE ENTRADA.
0050 ************************************************************
0060 * ---------------  DEFINICION DE VISTAS --------------------*
0070 DEFINE DATA LOCAL
0080 01 VIEW-ART-COM VIEW OF ARTICULOS-COMESTIBLES
0090       02 CVE-ART
0100       02 DESCRIP
0110       02 CANT
0120 * ----------------- DEFINICION DEL MAPA --------------------*
0130 01 ÑWA-MAPA
0140       02 ÑWA-CVE-ART                    (A7)
0150       02 ÑWA-DESCRIP                    (A30)
0160       02 ÑWN-CANT                (N5)
0170 END-DEFINE
0180 ****************************************
0190 **         INICIO DE PROGRAMA     **
0200 ****************************************
0210 REPEAT
0220     INPUT USING MAP 'ARTIC001'
0230     DECIDE ON FIRST VALUE OF *PF-KEY
0240             VALUE  'ENTR'
0250                 IF ÑWA-CVE-ART = ' ' THEN
```

```
0260                    REINPUT 'POR FAVOR DIGITE CLAVE DE
                        ARTICULO'
0270                    MARK *ÑWA-CVE-ART
0280                END-IF
0290                IF ÑWA-DESCRIP = ' ' THEN
0300                    REINPUT 'POR FAVOR DIGITE DESCRIPCION DE
                        ARTICULO'
0310                    MARK *ÑWA-DESCRIP
0320                END-IF
0330                IF ÑWN-CANT NE MASK(NNNNN) THEN
0340                    REINPUT 'DIGITAR CANTIDAD'
0350                    MARK *ÑWN-CANT
0360                END-IF
0370                PERFORM GRABA
0380            VALUE 'PF3'
0390                ESCAPE BOTTOM
0400            NONE
0410                IGNORE
0420    END-DECIDE
0430 END-REPEAT
0440 *
0450 ************************
0460 DEFINE SUBROUTINE GRABA
0470 ************************
0480 MOVE ÑWA-CVE-ART TO CVE-ART
0490 MOVE ÑWA-DESCRIP TO DESCRIP
0500 MOVE ÑWN-CANT    TO CANT
0510 STORE VIEW-ART-COM
0520 END TRANSACTION
0530 END-SUBROUTINE           /* GRABA
0540 ****************************************
0550 **        FIN DE PROGRAMA               **
0560 ****************************************
0570 END
```

5.4. Area local de datos

El Area Local de Datos contiene la definición de las variables a utilizar en un programa. Las variables locales pueden crearse de dos formas:

- Listando cada una de las variables dentro del Area de Definición de Datos (DEFINE DATA) o bien,
- Generando un Area Local fuera del programa y llamándola por medio de una sentencia especial dentro de DEFINE DATA. Se recomienda generar un Area Local de Datos principalmente para documentación, facilidad de lectura del programa, o bien, cuando varios programas utilizan exactamente las mismas variables.

El comando Natural para Crear/Editar un Area Local se encuentra descrita en el capítulo "Editores".

La forma de referenciar un Area Local de Datos dentro de un programa es la siguiente:

```
DEFINE DATA
      [LOCAL {USING área_de_datos}]
...
...
END-DEFINE
```

[LOCAL {USING área_de_datos}]

LOCAL indica que a continuación se hará la definición de variables locales.

USING área_de_datos se refiere al nombre del Area Local de Datos.

Si la cláusula USING es omitida, significa que la definición de variables locales se hará directamente en el programa, sin llamar un Area Local de Datos.

En un programa puede hacerse referencia a una o varias Areas Locales de Datos y también definir variables locales directamente.

Ejemplo :

```
0010 ************************************************************
0020 *                EJEMPLO DE AREA DE DATOS LOCAL
0030 *FUNCION: LLAMAR UN AREA DE DATOS LOCAL Y DEFINIR A SU VEZ
0040 *         VARIABLES LOCALES DENTRO DEL PROGRAMA
0050 ************************************************************
0060 DEFINE DATA LOCAL USING V01003
0070 LOCAL
0080 * --------------- DEFINICION DE VISTAS --------------------*
0090 01 VIEW-ART-COM VIEW OF ARTICULOS-COMESTIBLES
0100       02 CVE-ART
0110       02 DESCRIP
0120       02 CANT
0130 01 ÑWA-FECHA              (A08)
0140 END-DEFINE
0150 ****************************************
0160 **        INICIO DE PROGRAMA     **
0170 ****************************************
0180 MOVE *DATN TO ÑWA-FECHA   /*VARIABLE DEF. DENTRO DEL PROG.
0190 IF ÑWA-FECHA > '19970226' THEN
0200     READ VIEW-ART-COM BY CVE-ART
0210        ACCEPT IF CANT > 10
0220           MOVE CVE-ART TO ÑWA-CVE-ART   /* VARIABLES DEFINIDAS
0230                                         /* EN EL AREA V01003
0240           MOVE DESCRIP TO ÑWA-DESCRIP       /*
0250           MOVE CANT TO ÑWN-CANT             /*
```

```
0260          DISPLAY ÑWA-CVE-ART
0270                  ÑWA-DESCRIP
0280                  ÑWN-CANT
0290     END-READ
0300 END-IF
0310 ****************************************
0320 **        FIN DE PROGRAMA                    **
0330 ****************************************
0340 END
```

5.5. Area global de datos

Cuando se requiere que varios programas compartan los valores de un conjunto de variables, es necesario definir un Area Global de Datos. Esto es, los valores de las variables definidas en un área global pueden ser referenciadas a través de toda la aplicación.

Un Area Global de Datos también funciona como una forma de comunicación compartiendo variables entre un programa y una subrutina externa.

Un Area Global de Datos se crea como se indica en el capítulo "Editores".

El Area Global de Datos y los programas que hacen referencia a ella, deben definirse dentro de la misma librería.

Dentro de un programa, se declara un Area Global de Datos como sigue:

```
DEFINE DATA
        [GLOBAL USING data-área [WITH block[,block]]...]
...
...
END-DEFINE
```

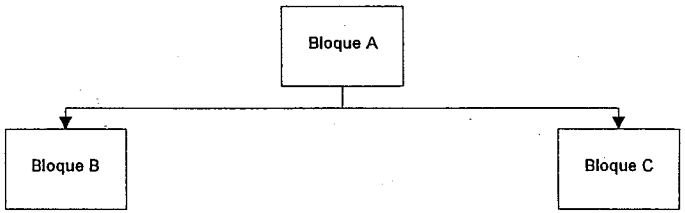
GLOBAL USING data-área

Especifica el nombre del Area Global a utilizar (8 caracteres). La cláusula GLOBAL puede especificarse sólo una vez dentro de la sentencia DEFINE DATA.

WITH block[,block]

Un bloque de datos es una colección de variables y/o vistas, son particularmente útiles en crear estructuras las cuales permiten que ciertos bloques de datos sean sobrescritos durante la ejecución de un programa, conservando de esta forma espacio de almacenamiento.

Por ejemplo, dada la siguiente estructura, los bloques B y C serían asignados a la misma área de almacenamiento. Esto implica que no es posible que los bloques B y C estén en uso al mismo tiempo.



A continuación se muestra un ejemplo de Area de Global de Datos conteniendo bloques:

```
9:00:01 **** E D I T   D A T A  *****                        97-03-12
Library: VENTAS   Name: ARTICOM  GLOBAL DBID: 10   FNR:     5
Command:                                           >        +
I T L    Name              F        Leng     Index/Init/EM/Name/Comment
- - - ---- all ------ -    ----     --------------------------
  B     BLOQUE-PRINCIPAL
    1   BP-CVE-ART         A        7
  B     SUB-BLOQUE1                                  BLOQUE-PRINCIPAL
    1   SB1-CVE-PRESNT     A        5
  B     SUB-BLOQUE2                                  BLOQUE-PRINCIPAL
    1   SB2-CANT  N        5
```

Una forma de invocar el Area Global del ejemplo es:

Programa 1:

```
DEFINE DATA
     . GLOBAL USING ARTICOM
```

```
END-DEFINE
```

Programa 2:

```
DEFINE DATA
    GLOBAL USING ARTICOM
     WITH BLOQUE-PRINCIPAL.SUB-BLOQUE1
END-DEFINE
```

5.6 Rutinas de Ayuda (Helproutine)

Una Rutina de Ayuda, es un conjunto de instrucciones que serán invocadas con la finalidad de proporcionar información de ayuda al usuario durante la ejecución de un mapa (Help Global para el mapa), o bien, puede ser el nombre de una rutina de ayuda asignada a un campo.

Los pasos para el manejo de una Rutina de Ayuda son :

- Crear la rutina de ayuda,
- Invocar la rutina de ayuda desde el programa,
- Declarar la rutina de ayuda en el mapa.

Crear la rutina de ayuda

Una rutina de ayuda se crea por medio del Editor de Programas de Natural como se ha descrito en el capítulo "Editores".

En el editor se captura el programa que genera la rutina de ayuda a desplegar, incluyendo las validaciones a posibles campos de entrada, así como paginación de texto si se necesita.

Otra forma de crear una rutina de ayuda es por medio del Editor de Mapas de Natural, seleccionando la opción H (Initialize a New Help Text). Esta opción se emplea cuando se crea un texto global para un mapa.

Si la rutina de ayuda devuelve algún parámetro al programa que la invoca, entonces la Helproutine comenzará con la sentencia DEFINE DATA PARAMETER como ese explica en el siguiente paso.

Invocar la rutina de ayuda desde el programa

La Helproutine (HE) es un Parámetro de Sesión de Natural que se procesa en combinación con la instrucción INPUT o durante la ejecución de un mapa posicionando el cursor bajo un campo y presionando la tecla de función de ayuda (definida dentro del programa mediante la instrucción SET KEY), o digitando el caracter de ayuda (' ? ' por default) en el campo.

La sintaxis de este parámetro es:

HE = operando1 [, {operando2}]
=

Operando1

El nombre de la rutina de ayuda que será invocada. Este puede ser especificado como una constante de texto o como una variable definida por el usuario que contiene el nombre de la rutina.

Operando2

Un parámetro que puede ser pasado a la rutina de ayuda. Este puede ser una constante o una variable definida por el usuario, la cual contiene el valor del parámetro. Si se especifica ' = ', el nombre del campo como está definido en el mapa será pasado a la rutina de ayuda. En el caso de una rutina de ayuda la cual es asignada a un mapa, ' = ' se refiere al nombre del mapa.

Si se especifica un parámetro, la rutina de ayuda deberá comenzar con una sentencia DEFINE PARAMETER, la cual define un campo que corresponde en formato y longitud con el parámetro.

Si se utiliza la notación de parámetro ' = ' para pasar un campo o nombre de mapa, el parámetro en la rutina de ayuda ***debe*** ser definido como ' A65 '.

El valor del campo para el cual una rutina de ayuda es especificada puede ser referenciado dentro de la Rutina de Ayuda. Esto se hace definiendo un campo en la instrucción DEFINE PARAMETER la cual corresponde en formato y longitud con el campo original. En el bloque de campos dentro de la instrucción DEFINE PARAMETER, este campo siempre ***debe*** ser definido como el último.

Si una Rutina de Ayuda se invoca digitando ' ? ' en el campo o por medio de la tecla de función 'PF1', los datos restantes que pueden ser ingresados en otros campos *no* son asignados a las variables del programa hasta que el requerimiento de ayuda haya sido procesado.

Ejemplo:

```
0010 /* PROGRAMA PRINCIPAL
0020 DEFINE DATA
0030 01 ÑWA-CVE-ART           (A7)
0040 END-DEFINE
0050 INPUT ÑWA-CVE-ART (HE='AYUDA')
0120 END


0010 /* HELP ROUTINE ' AYUDA '
0020 DEFINE DATA PARAMETER
0030 01 ÑWA-CVE-ART           (A7)
0040 END-DEFINE
...
...
```

Declarar la rutina de ayuda en el mapa

Este paso es necesario sólo si se invoca un mapa dentro del programa principal.

Para declarar una rutina de ayuda de un campo, entrar al Editor de Mapas e invocar el mapa correspondiente. A continuación, ubicarse en la primera posición del campo en el cual se quiere desplegar la ayuda y digitar '.E' (Editar). Aparecerá una pantalla donde se definen las características del campo, entonces ubicarse en HE _____ (Help Routine). Se recomienda digitar ' + ' para capturar el nombre de la rutina de ayuda y el parámetro, en cuyo caso se desplegará una pequeña pantalla solicitando estos datos. También se puede teclear directamente en la línea, pero se corre el riesgo de no contar con suficiente espacio para escribir el nombre y el parámetro.

5.7 Copia de Código

Una Copia de Código (Copycode), es un segmento de código que será incluido dentro de un programa NATURAL. La forma de crear un Copycode ha sido descrita en el capítulo "Editores".

Un Copycode no tiene sección DEFINE DATA, únicamente instrucciones NATURAL.

En el siguiente ejemplo, se crea un CopyCode con el nombre ARTCOM01 :

```
0010 * EJEMPLO DE DEFINICION DE COPYCODE
0020 READ VIEW-ART-COM BY CVE-ART
0030           ACCEPT IF CANT > 0
0040                  DISPLAY  CVE-ART  CANT   PRECIO-PUBLICO
0050                  ADD CANT TO ÑWN-CANT-TOT
0060 END-READ


0010 * USO DE COPYCODE-PROGRAMA PRINCIPAL
0020 DEFINE DATA LOCAL USING V0120M03 /* Area Local,  contiene
0030                                  /*  la  vista  de  Artículos
Comestibles
0040 01 ÑWN-CANT-TOT          (N7)
0050 END-DEFINE
0060 ****************************************
0070 **        INICIO DE PROGRAMA            **
0080 ****************************************
0090 INCLUDE ARTCOM01
0100 FETCH 'PROGART2'  ÑWN-CANT-TOT
0110 ...
0120 ****************************************
0130 **        FIN DE PROGRAMA               **
0140 ****************************************
0150 END
```

La instrucción INCLUDE es la sentencia NATURAL que permite referenciar un Copycode dentro de un programa. Su formato es el siguiente:

INCLUDE nombre_copycode [{librería}]

Las líneas de código dentro del Copycode, no serán físicamente incluidas en el programa, sino el programa objeto resultante del proceso de compilación. Por tanto, cuando un Copycode es modificado, el programa que hace referencia a éste, deberá ser recompilado.

Un mismo Copycode no puede ser utilizado dos veces en el mismo programa.

5.8 Llamado a otros programas

Al principio de este capítulo se mencionó la necesidad de dividir una función en varias partes con fines de mantener la modularidad en un sistema. Por ello dentro de un programa se requiere invocar frecuentemente a otros programas.

Un programa puede llamar a otro dentro del lenguaje Natural por medio de la siguiente sentencia:

FETCH [{RETURN}] nombre_programa parámetros
[{RETURN]

Sin la especificación de RETURN, la ejecución del programa que invoca terminará inmediatamente y el programa invocado será activado como el programa principal (nivel 1).

Si se especifica RETURN, la ejecución del programa que ejecutó el FETCH será suspendida y el programa invocado será activado como un subprograma en un alto nivel. El control es regresado al programa que ejecutó el FETCH RETURN cuando una instrucción END o ESCAPE ROUTINE sea encontrado en el programa invocado. El control es regresado a la siguiente instrucción después del FETCH RETURN.

FETCH RETURN permite llamar un programa como una subrutina, la cual ha sido escrita como un programa principal. Además de los parámetros pasados explícitamente como parámetros del FETCH, el programa invocado también tiene acceso a la Area Global de Datos especificada.

Nombre_programa

Programa que se invoca, el cual deberá haber sido previamente catalogado.

Parámetros

Lista de los nombres de variables que serán pasados al programa invocado como parámetro. El programa invocado deberá tener definida el área PARAMETER dentro de la sección DEFINE DATA como se muestra a continuación:

```
DEFINE DATA
PARAMETER
  (lista de variables recibidas desde el programa que invoca)
...
...
END-DEFINE
```

Ejemplo :

```
0010 ************************************************************
0020 *                EJEMPLO DE LA INSTRUCCION FETCH
0030 *FUNCION: LLAMAR A UN PROGRAMA DENTRO DE OTRO PROGRAMA CON
0040 *                PASO DE PARAMETROS
0050 ************************************************************
0060 DEFINE DATA
0070 LOCAL
0080 01 MAPA-ARTICULOS                (A47)
0090 01 REDEFINE      MAPA-ARTICULOS
0100      02 CVE-ART                 (A07)
0110      02 NOMBRE-ART              (A10)
0120      02 DESCRIP                 (A30)
0130 END-DEFINE
0140 ***************************************
0150 **        INICIO DE PROGRAMA           **
0160 ***************************************
0170 REPEAT
0180    INPUT USING MAP NM5031
0190    DECIDE ON FIRST VALUE OF *PF-KEY
0200       VALUE 'ENTR'
0210         IF OPCION = '1' THEN
0220            FETCH RETURN 'PROG01' CVE-ART NOMBRE-
                  ART DESCRIP
0230         END-IF
0240         IF OPCION = '2' THEN
0250            FETCH RETURN 'PROG02' CVE-ART NOMBRE-
                  ART DESCRIP
0260       VALUE 'PF3'
0270         ESCAPE BOTTOM
0280       NONE
0290         IGNORE
0300    END-DECIDE
0310 END-REPEAT
0320 ***************************************
0330 **        FIN DE PROGRAMA                **
0340 ***************************************
0350 END
```

```
0010 ************************************************************
0020 *                       PROGRAMA PROG01
0030 *         (INVOCADO EN EL EJEMPLO ANTERIOR)
0040 ************************************************************
0050 DEFINE DATA LOCAL
0060 01 CVE-ART                      (N07)          /*  PARAMETROS
0070 01 NOMBRE-ART            (A10)
0080 01 DESCRIP                      (A30)
0090 END-DEFINE
0100 ********************************************
0110 **        INICIO DE PROGRAMA          **
0120 ********************************************
0130 IF *DATA > 0
0140       INPUT CVE-ART NOMBRE-ART DESCRIP  /* RECIBE PARAMETROS
0150 END-IF
...
...
0190 ****************************************
0200 **        FIN DE PROGRAMA                    **
0210 ****************************************
0220 END
```

6. VISTAS, ACCESOS Y MANIPULACION A LA BASE DE DATOS.

6.1 Vistas de usuarios.

Como un antecedente al concepto de vistas de usuarios, es necesario definir primeramente qué es un Módulo de Definición de Datos (DDM).

Una DDM es la definición de un módulo en una base de datos, es decir, es una copia de la estructura de un archivo y contiene las características de cada campo, tales como nombre, longitud, nombre abreviado, formato fijo o variable, si es descriptor, etc.

Las vistas de usuario son una forma de visualizar los registros de un archivo definido en base al Módulo de Definición de Datos (DDM), además de ser la estructura lógica por medio de la cual se manejará la información de los campos de cada registro accesado mediante esta vista.

Estas vistas pueden ser declaradas en el área de definición de datos del programa, subprograma o de manera externa en un área local de datos. Se recomienda definirlas en una área local para tener modularidad en la aplicación y además porque es muy posible que las vistas definidas en el área local externa sean requeridas por algún otro objeto NATURAL.

Las vistas creadas para ser utilizadas en la actualización de un archivo de la base de datos, no deben incluir superdescriptores, ni subdescriptores. Las vistas que son utilizadas para la lectura de algún archivo de la base de datos, sí pueden contener en su definición algún superdescriptor o subdescriptor.

Ejemplo :

De la DDM ARTICULOS-COMESTIBLES generar una vista y llamarla VIEW-ART-COM.

DDM: ARTICULOS-COMESTIBLES

```
TY  L   DB  NAME                 F       LENG    S   D  REMARKS
--  -   --  -------------------  -       ----    -   -  -------
    1   AA  CVE-ART              A          7        D
            HD=CLAVE DE ARTICULO
    1   AB  NOMBRE-ART           A         10        D
            HD=NOMBRE DEL ARTICULO
    1   AC  CVE-PRESNT           A          5        D
            HD=CLAVE DE PRESENTACION
    1   AD  DESCRIP              A         30        N
            HD=DESCRIPCION DEL PRODUCTO
    1   AE  CANT                 N          5
            HD=CANTIDAD EN EXISTENCIA
G   1   AF  PRECIO-VENTA
            HD=PRECIO DE VENTA
    2   AG  PRECIO-PUBLICO       N        6.2
            HD=PRECIO AL PUBLICO
    2   AH  PRECIO-DISTRIBUIDOR  N        6.2
            HD=PRECIO AL DISTRIBUIDOR
M   1   AI  CVE-DISTRIBUIDOR     A          5
            HD=CLAVE DEL DISTRIBUIDOR
P   1   AJ  PERIODO-CONSUMO
    2   AK  FECHA-PRODUCCION                        D
            HD=FECHA DE PRODUCCION
    2   AL  FECHA-CADUCIDADM                        D
            HD=FECHA DE CADUCIDAD
    2   AM  CANT-PROD            N          5
    1   S1  SP-CVE-ART-PRESNT    A         12        S
            CVE-ART (A7)
            CVE-PRESNT (A5)
```

VISTA: VIEW-ART-COM.

```
0010 DEFINE DATA  LOCAL
0020 01 VIEW-ART-COM VIEW OF ARTICULOS-COMESTIBLES
0030        02 CVE-ART
0040        02 NOMBRE-ART:
0050        02 CVE-PRESNT
0060        02 DESCRIP
0070        02 CANT
0080        02 PRECIO-PUBLICO
0090        02 PRECIO-DISTRIBUIDOR
0100        02 CVE-DISTRIBUIDOR (1:10)
0110        02 PERIODO-CONSUMO  (1:10)
0120            03 FECHA-PRODUCCION
0130            03 FECHA-CADUCIDAD
0140            03 CANT-PROD
0150        02 SP-CVE-ART-PRESNT
0160 END-DEFINE
```

6.2 Accesos complejos
6.2.1 ESCAPE

Sintaxis

```
ESCAPE { TOP
         BOTTOM [(r) ] [IMMEDIATE]
         ROUTINE [IMMEDIATE] }
```

El postulado **ESCAPE** interrumpe el flujo lineal de la ejecución de un ciclo de proceso o una rutina.

Las palabras reservadas **TOP, BOTTOM** y **ROUTINE** indican donde el proceso continuará, después de ser encontrado un postulado **ESCAPE.**

Un postulado **ESCAPE TOP/BOTTOM,** cuando es encontrado en un proceso, hace referencia al ciclo más interno.

ESCAPE TOP

TOP indica que el proceso continuará al principio del ciclo. La siguiente repetición del ciclo es iniciada.

ESCAPE BOTTOM

BOTTOM significa que el ciclo de proceso continuará con el primer postulado siguiente al ciclo de proceso, dentro del cual éste es invocado.

Si BOTTOM es seguido por una etiqueta o número de referencia, el proceso continuará en el primer postulado siguiente al ciclo de proceso identificado por la etiqueta o número de referencia.

Si se añade la opción IMMEDIATE, ningún fin de ciclo será ejecutado.

ESCAPE ROUTINE

Esta opción indica que la actual rutina NATURAL invocada vía PERFORM, CALLNAT, FETCH RETURN o el mismo programa principal, terminará su ejecución y cederá el control a quien la invocó.

En el caso de un proceso invocado por un PERFORM, CALLNAT o FETCH RETURN, el proceso continuará con el primer postulado después de cada uno de éstos. En el caso del programa principal, el control será recibido por el procesador de comandos NATURAL.

Todos los ciclos actualmente activos serán terminados y los procesos de fin de ciclo serán ejecutados .

Si se especifica la opción IMMEDIATE, ningún fin de ciclo será ejecutado.

Si uno de estos postulados es encontrado durante el proceso de una condición IF-NO-RECORD-FOUND, ningún proceso de fin de ciclo será ejecutado..

Ejemplo de ESCAPE:

```
0010 DEFINE DATA LOCAL
0020 01 VIEW-ART-COM VIEW OF ARTICULOS-COMESTIBLES
0030    02 CVE-ART
0040    02 DESCRIP
0050    02 CANT
0060    02 PRECIO-PUBLICO
0070 01 #ARTICULO    (A20)
0080 01 #CNTL        (A01)
0090 END-DEFINE
0100 ************************************
0110 **      INICIO DE PROGRAMA
0120 ************************************
0130 RPT. REPEAT
0140       INPUT 'Introduzca el nombre de un artículo :'
#ARTICULO
0150       /  '( o  .  para terminar.)'
0160       IF #ARTICULO = '.'
0170         STOP
0180       END-IF
0190 FND. FIND VIEW-ART-COM WITH CVE-ART = #ARTICULO
0200          IF NO RECORDS FOUND
0210             WRITE ' Artículo no encontrado'
0220             ESCAPE BOTTOM(FND.)
0230          END-NOREC
0240          AT START OF DATA
0250            INPUT (AD=O) 'Registros encontrados:'  *NUMBER //
0260               'Digitar MDM para desplegar datos' #CNTL
                   (AD=MT'_')
0270               IF #CNTL EQ 'D'
0280                  ESCAPE BOTTOM(FND.)
0290               END-IF
0300          END-START
0310          DISPLAY NOTITLE DESCRIP CANT PRECIO-PUBLICO
0320       END-FIND
0330 END-REPEAT
0340 ************************************
0350 **      FIN DE PROGRAMA
0360 ************************************
0370 END
```

6.2.2 Notación de referencia de postulados

Una etiqueta o un número de línea de código fuente pueden ser usados para referirse a una instrucción NATURAL, además de ser utilizados con propósitos de documentación.

Cualquier instrucción en NATURAL que cause un ciclo de proceso puede ser marcada con una etiqueta simbólica para referencias subsecuentes. Una etiqueta puede también ser especificada de la forma (LABEL.) después del objeto referenciado.

Ejemplo :

```
FND. FIND VIEW-ART-COM WITH CVE-ART = #CVE-ART
          MOVE #NUEVO-NOM TO NOMBRE-ART
          UPDATE(FND.)
     END-FIND
```

Un postulado puede ser referenciado usando el número de línea de código fuente. Los cuatro dígitos del número de línea deben de ser especificados.
Ejemplo :

```
0080 FIND VIEW-ART-COM WITH CVE-ART = #CVE-ART
0090           MOVE #NUEVO-NOM TO NOMBRE-ART
0100           UPDATE(0080)
0110 END-FIND
```

6.2.3 ACCEPT

Sintaxis

```
ACCEPT/REJECT { TOP                          } [IF] CONDICION-
              { BOTTOM [(r)] [IMMEDIATE]     }  LOGICA
              { ROUTINE IMMEDIATE            }
```

Las instrucciones ACCEPT y REJECT son usados para aceptar o rechazar un registro basado en un criterio lógico especificado por el usuario en conjunción con las instrucciones FIND, READ, HISTOGRAM, CALL FILE, SORT, o READ WORK FILE que realizan lectura de registros de datos en ciclos de proceso. El criterio es evaluado después de que los registros han sido seleccionados/leídos.

Si una instrucción ACCEPT/REJECT es encontrada en un proceso, éste internamente referenciará al ciclo de proceso actualmente activo más interno, dicho ciclo de proceso debe ser realizado con alguna de las instrucciones arriba mencionadas.

Los campos de la condición lógica pueden ser campos de la base de datos o variables definidas por el usuario. Cuando ACCEPT/REJECT se utiliza en combinación con la instrucción HISTOGRAM, sólo los campos de la base de datos deben incluirse como criterio lógico.

Ejemplo :

```
0010 DEFINE DATA LOCAL
0020 01 VIEW-ART-COM VIEW OF ARTICULOS-COMESTIBLES
0030    02 CVE-ART
0040    02 NOMBRE-ART
0050    02 CVE-PRESNT
0060    02 CANT
0070 END-DEFINE
0080 ************************************
0090 **       INICIO DE PROGRAMA
0110 ************************************
0120 READ VIEW-ART-COM BY CVE-ART
0130      ACCEPT IF CANT > 0
0140           WRITE NOTITLE  '=' NOMBRE-ART '=' CVE-PRESNT
0150 END-READ
0160 ************************************
0170 **       FIN DE PROGRAMA
0180 ************************************
0190 END
```

6.2.4 SORT

Sintaxis :

```
END-ALL
[AND]
SORT [ { THEM    } ] [BY] { Operando1 [ { ASCENDING  } ] }
     [ { RECORDS } ]      {           [ { DESCENDING } ] }
                          {                              }

     USING-cláusula
     [GIVE-cláusula]
     instrucciones

END-SORT
```

La instrucción SORT ejecuta una operación de ordenación de campos, se puede hacer de dos modos: con el programa de ordenación provisto por el sistema operativo (modo batch) o el de ordenación en línea. Ambos ordenan los registros de todos los ciclos de proceso que están activos cuando la instrucción SORT es ejecutada.

Sólo una instrucción SORT es permitida por programa NATURAL (válida únicamente dentro del código principal). Si se necesitan varias secuencias de ordenación para diferentes reportes, los datos deben ser escritos a un WORK FILE y entonces escribir instrucciones SORT por separado en programas NATURAL. Hasta 10 campos pueden ser referenciados con el criterio de ordenado en la instrucción SORT. La longitud de los registros dentro del SORT no debe exceder 4096 bytes.

Un JCL adicional es requerido cuando la instrucción SORT se usa en modo batch.

El operando1 representa los campos/variables del criterio de ordenado y pueden incluirse de 1 a 10 campos de la base de datos. Un campo múltiple o periódico son válidos, pero un grupo o arreglos no son permitidos.

La longitud combinada de todos los campos referenciados en el criterio de ordenado no puede ser mayor a 253 bytes.

ASCENDING causa que una operación sea ejecutada en orden ascendente, en tanto que DESCENDING implica el ordenamiento en forma descendente. La secuencia por default es ASCENDING. ASCENDING/DESCENDING pueden ser especificadas para cada campo a ordenar.

Cláusula USING { USING operando2... | USING KEYS }

La cláusula USING indica los campos a ser escritos en el área de almacenamiento intermedio de sort. USING KEYS causa que los campos por medio de los cuales se va a ordenar sean escritos en el área intermedia de ordenado.

USING Operando2 incluye campos adicionales además de los especificados en USING KEYS.

Si después de la ejecución del sort se hace referencia a un campo el cual no fue escrito en el área de almacenamiento intermedio de sort, el valor del campo será el último valor del campo antes de la clasificación.

Ejemplo :

```
0180   READ VIEW-ART-CON WITH CVE-ART = #CVE-ART
0190   END-ALL
0200   SORT BY CVE-ART CVE-PRESNT
0210        USING PRECIO-VENTA PRECIO-PUBLICO
               PRECIO-DISTRIBUIDOR
0220   /* Se muestra la información ordenada
0220        DISPLAY PRECIO-VENTA  PRECIO-PUBLICO
               PRECIO-DISTRIBUIDOR
0230   END-SORT
```

6.3 Instrucciones de actualización

Las instrucciones de actualización que tenemos en NATURAL son las siguientes:

6.3.1 STORE

Adiciona un registro a un archivo de base de datos.

Sintaxis :

```
STORE [ RECORD] [IN] [FILE]  nombre-vista
        [PASWORD = operando1]
        [CIPHER    = operando1]
```

[[[USING | GIVING]] NUMBER operando2] [(r)]

Las cláusulas PASWORD y CIPHER proporcionan una palabra de paso o una clave cuando el archivo se encuentra protegido.

La cláusula USING/GIVING adiciona un registro con un ISN proporcionado por el usuario. Si ya existe un registro con el ISN especificado, un mensaje de error será regresado y la ejecución del programa terminará a menos que se incluya la instrucción ON ERROR.

6.3.2 UPDATE

Actualiza los campos de un registro. El registro debe haber sido previamente seleccionado por medio de las instrucciones FIND, GET, o READ.

Sintaxis :

UPDATE [RECORD] [IN] [STATEMENT] [(r)]

La referencia a la instrucción (r), indica la instrucción en la cual el registro a ser modificado fue leído. Si (r) se omite, la instrucción UPDATE referenciará el ciclo de lectura activo más interno, (r) puede ser un número de referencia de instrucción o una etiqueta de instrucción.

6.3.3 DELETE

Esta instrucción elimina registros de una base de datos.

Sintaxis :

DELETE [RECORD] [IN] [STATEMENT] [(r)]

La referencia a la instrucción (r) es equivalente a la descrita en la instrucción UPDATE.

6.3.4 END TRANSACTION

END TRANSACTION indica el fin de una transacción lógica. Una transacción lógica es una unidad lógica de trabajo que debe de ser procesada en su totalidad para garantizar que la información contenida en la base de datos sea lógicamente consistente.

Ejecuciones sucesivas de una instrucción END TRANSACTION aseguran que todas las actualizaciones sean físicamente realizadas en la base de datos. Si una instrucción END TRANSACTION no termina exitosamente, las actualizaciones son automáticamente ignoradas.

Después de ejecutar esta instrucción son liberados los registros retenidos (en status de HOLD).

Sintaxis :

END [OF] TRANSACTION

6.3.5 BACKOUT TRANSACTION

Esta instrucción ignora todas las actualizaciones realizadas por el usuario.

Después de ejecutar esta instrucción son liberados los registros retenidos (en status de HOLD).

Sintaxis :

BACKOUT TRANSACTION

Ejemplo de la instrucción STORE:

```
0010 DEFINE DATA LOCAL
0020 01 VIEW-ART-COM VIEW OF ARTICULOS-COMESTIBLES
0030    02 CVE-ART
0040    02 NOMBRE-ART
0050    02 CVE-PRESNT
0060    02 DESCRIP
0070    02 CANT
0080 01 #M-CVE-ART                    (A7)
0090 01 #M-NOMBRE-ART                 (A10)
0100 01 #M-CVE-PRESNT                 (A5)
0110 01 #M-DESCRIP                   (A30)
0120 01 #M-CANT                      (N5)
0130 01 #M-PRECIO-PUBLICO             (N6.2)
0140 END-DEFINE
0150 ****************************************
0160 **      INICIO DE PROGRAMA
0170 ****************************************
0180 INPUT USING MAP 'P35DP450' /* Se invoca mapa donde se piden
0190                          / * los datos de un nuevo registro a
adicionar
0200    /* Se mueven los datos capturados a los campos de la vista
0210 MOVE #M-CVE-ART                 TO CVE-ART
0220 MOVE #M-NOMBRE-ART              TO NOMBRE-ART
0230 MOVE #M-CVE-PRESNT              TO CVE-PRESNT
0240 MOVE #M-DESCRIP                 TO DESCRIP
0250 MOVE #M-CANT                    TO CANT
0260 MOVE #M-PRECIO-PUBLICO TO PRECIO-PUBLICO
0270 STORE VIEW-ART-COM                  /* Se dá de alta el reg.
0280 END TRANSACTION                     /* Efectúa físicamente la
                                            actualización
0290 ****************************************
0300 **      FIN DE PROGRAMA
0310 ****************************************
0320 END
```

000061

Ejemplo de la instrucción UPDATE:

```
0010 DEFINE DATA LOCAL
0020 01 VIEW-ART-COM VIEW OF ARTICULOS-COMESTIBLES
0030        02 CVE-ART
0040        02 CVE-PRESNT
0050 01 #M-CVE-ART                  (A7)
0060 01 #M-CVE-PRESNT               (A5)
0070 END-DEFINE
0080 ****************************************
0090 **       INICIO DE PROGRAMA
0100 ****************************************
0110 INPUT USING MAP 'P35DP700' /* Se invoca mapa donde se pide
0120             / * el nuevo tipo de presentación para un artículo
0130 FIND VIEW-ART-COM WITH CVE-ART EQ #M-CVE-ART
0140          MOVE #M-CVE-PRESNT    TO  CVE-PRESNT
0150          UPDATE(0100)    /* Se actualiza el registro
0160          END TRANSACTION /* Efectúa físicamente la
actualización
0170 END-FIND
0180 ****************************************
0190 **       FIN DE PROGRAMA
0200 ****************************************
0210 END
```

Ejemplo de la instrucción DELETE:

```
0010 DEFINE DATA LOCAL
0020 01 VIEW-ART-COM VIEW OF ARTICULOS-COMESTIBLES
0030        02 CVE-ART
0040 01 #M-CVE-ART                  (A7)
0050 END-DEFINE
0060 ****************************************
0070 **       INICIO DE PROGRAMA
0080 ****************************************
0090 INPUT USING MAP 'P35DP900' /* Se invoca mapa donde se pide
0100             / * la clave del artículo que se quiere borrar
0110 FIND VIEW-ART-COM WITH CVE-ART EQ #M-CVE-ART
0120          DELETE(0110)    /* Se borra el registro
0130          END TRANSACTION /* Efectúa físicamente la actualización
0140 END-FIND
0150 ****************************************
0160 **       FIN DE PROGRAMA
0170 ****************************************
0180 END
```

6.4 Accesos a archivos secuenciales.

Los datos de archivos secuenciales (que no son archivos ADABAS), se accesan mediante las instrucciones READ WORK FILE y WRITE WORK FILE con programas que se ejecutan en modo BATCH.

6.4.1 READ WORK FILE.

Sintaxis:

READ WORK [FILE] work-file-number
END-WORK

Función:

Esta instrucción se utiliza para leer registros de archivos físicos secuenciales . Causa un ciclo de lectura de todos los registros del archivo.

work-file-number:

Es el número del archivo de trabajo a ser leído (como está definido para NATURAL).

Ejemplo:

```
0010 DEFINE DATA LOCAL
0020 01 VIEW-ART-COM VIEW OF ARTICULOS-COMESTIBLES
0030        02 CVE-ART
0040        02 NOMBRE-ART
0050        02 DESCRIP
0060 01 ÑWORK-FILE-ARTICULOS
0070        02 ÑW-NOMBRE-ART                (A7)
0080        02 ÑW-DESCRIP                   (A30)
0090 END-DEFINE
0100 ****************************************
0120 **       INICIO DE PROGRAMA
0130 ****************************************
0140 READ WORK FILE 1 ÑWORK-FILE-ARTICULOS
0150       WRITE NOTITLE  '=' ÑW-NOMBRE-ART '=' ÑW-DESCRIP
0160 END-WORK
0170 ****************************************
0180 **       FIN DE PROGRAMA
0190 ****************************************
0200 END
```

000062

6.4.2. WRITE WORK FILE.

Sintaxis:

WRITE WORK [FILE] work-file-number [VARIABLE] [operand]...

Función:

Sentencia usada para escribir registros a archivos físicos secuenciales.

work-file-number:

Es el número del archivo de trabajo para escritura.

VARIABLE:

Los campos o el nombre del grupo de campos a ser escritos en el archivo secuencial.

Ejemplo:

```
0010 DEFINE DATA LOCAL
0020 01 VIEW-ART-COM VIEW OF ARTICULOS-COMESTIBLES
0030         02 CVE-ART
0040         02 NOMBRE-ART
0050         02 CVE-PRESNT
0060         02 DESCRIP
0070         02 CANT
0080 01 ÑWORK-FILE-ARTICULOS
0090         02 ÑW-CVE-ART              (A7)
0100         02 ÑW-NOMBRE-ART           (A10)
0110         02 ÑW-CVE-PRESNT           (A5)
0120         02 ÑW-CANT                 (N5)
0130 END-DEFINE
0140 ************************************************
0150 **       INICIO DE PROGRAMA
0160 ************************************************
0170 READ VIEW-ART-COM BY CVE-ART
0180       MOVE VIEW-ART-COM TO ÑWORK-FILE-ARTICULOS
0190       WRITE WORK FILE 1 ÑWORK-FILE-ARTICULOS
0200 END-READ
0210 ************************************************
0220 **       FIN DE PROGRAMA
0230 ************************************************
0240 END
```

7. MANIPULACION DE DATOS.

En este capítulo se describirá y ejemplificará el uso de estructuras de datos y sentencias Natural especiales relacionadas con el manejo de datos.

En ocasiones es necesario convertir campos de un formato a otro, saber el número máximo de ocurrencias ocupadas de un campo múltiple, almacenar y recuperar información en un arreglo, realizar operaciones aritméticas con campos de fecha y realizar manipulaciones de campos alfanuméricos.

7.1 Definición y manejo de arreglos.

Un arreglo es una estructura de datos que contiene un único identificador para representar a múltiples datos individuales, pudiendo referenciarse a cada uno por separado. Así un arreglo de una dimensión puede considerarse como una lista de datos del mismo tipo. Un arreglo de dos dimensiones se visualiza como una tabla y un arreglo de tres dimensiones se maneja como una tabla de tablas.

Los siguientes son ejemplos de definición y manejo de arreglos en programas NATURAL.

a) Ejemplos de definiciones de arreglos.

- ÑP-ARRAY-03 (N3/1:10)

  Define un arreglo de una dimensión con 10 ocurrencias.

- ÑP-ARRAY-02 (A5/4,2)

  Define un arreglo de dos dimensiones con 4 y 2 ocurrencias para cada una de las dimensiones respectivamente .

- ÑP-ARRAY-01 (N8/1:10,1:5,1:3)

  Define un arreglo de tres dimensiones con 10, 5 y 3 ocurrencias para cada una de las dimensiones respectivamente .

b) Ejemplos de manejo de arreglos.

- ÑP-ARRAY-03 (ÑP-CONTADOR)

  Hace referencia a una ocurrencia que está determinada por el valor de la variable ÑP-CONTADOR.

- ÑP-ARRAY-03 (6:10)

  Hace referencia a las ocurrencias, desde la 6 hasta la 10 en un arreglo de una dimensión.

- ÑP-ARRAY-03 (*)

  Hace referencia a todas las ocurrencias de un arreglo de una dimensión.

- ÑP-ARRAY-03 (*,*,*)

  Hace referencia a todas las ocurrencias de un arreglo de tres dimensiones.

Ejemplo:

```
0010 ************************
0020 DEFINE DATA LOCAL
0030 ************************
0040 01 ARTICULOS VIEW OF ARTICULOS-COMESTIBLES
0050       02 CVE-ART                          (A7)
0060       02 CVE-PRESNT                       (A5)
0070       02 NOMBRE-ART                       (A10)
0080       02 C*CVE-DISTRIBUIDOR
0090       02 CVE-DISTRIBUIDOR                 (A5/1:10)
0100       02 CANT                             (N5)
0110 01 ÑP-CONT01                               (N6)
0120 01 ÑP-CONT02                               (N6)
0130 01 ÑP-ARRAY-ARTICULOS                       (A07/1:20)
0140 01 ÑP-ARRAY-DISTRIBUIDORES                 (N05/1:10)
0150 END-DEFINE
0160 READ (20) ART BY CVE-ART
0170       ADD 1 TO ÑP-CONT01
0180       MOVE CVE-ART TO ÑP-ARRAY-ARTICULOS(ÑP-CONT01)  /* guarda
           20 claves de artículos en el arreglo.
0190       FOR ÑP-CONT02  1  TO  10          /* guardar 10 valores
0200           MOVE CVE-DISTRIBUIDOR(ÑP-CONT02) TO
0210                ÑP-ARRAY-DISTRIBUIDORES(ÑP-CONT02)
0220         END-FOR
0230 END-READ
0240 END
```

7.2 Uso de la variable C*

En ocasiones es necesario conocer el número máximo de ocurrencia(s) ocupadas por un campo múltiple o por un campo periódico. Esto se puede saber mediante la definición del contador interno de ocurrencias (C*) de ADABAS. Este contador debe ser especificado inmediatamente antes del nombre del campo, cuyo formato interno es numérico y con longitud de 3 posiciones.

Ejemplo:

```
0010 ******************
0020 DEFINE DATA
0030 ******************
0040 LOCAL
0050 01 ARTICULOS VIEW OF ARTICULOS-COMESTIBLES
0060       02 CVE-ART                          (A7)
0070       02 CVE-PRESNT                       (A5)
0080       02 NOMBRE-ART                       (A10)
0090       02 C*CVE-DISTRIBUIDOR
0100       02 CVE-DISTRIBUIDOR                 (A5/1:10)
0110       02 CANT                             (N5)
0120 01 ÑP-CONT01                               (N6)
0120 01 ÑP-CONT02                               (N6)
0130 01 ÑP-ARRAY-ARTICULOS                       (A07/1:20)
0130 01 ÑP-ARRAY-DISTRIBUIDORES                 (N05/1:10)
0130 END-DEFINE
0140 READ ARTICULOS BY CVE-ART
0150      FOR ÑWN-CONT 1 TO C*CVE-DISTRIBUIDOR
0160           DISPLAY '='CVE-ART '='CVE-DISTRIBUIDOR(ÑWN-CONT)
0170      END-FOR
0180 END-READ
0190 END
```

7.3 Manejo de la instrucción MOVE.

La sentencia MOVE es usada para mover el valor de un operando a uno o más operandos (campos o arreglos), y dar un formato a otro formato utilizando máscaras de edición. Existen varios casos para la sentencia MOVE:

MOVE ROUNDED

Esta opción redondea un valor sólo si es numérico.

Ejemplo:

ÑP-ENTERO (N1)

MOVE ROUNDED 1.995 TO ÑP-ENTERO
La variable ÑP-ENTERO tendrá el valor 2.

MOVE BY NAME

Esta opción mueve campos individuales que están contenidos en una estructura de datos a otra estructura sin importar su posición. Sólo permite mover campos que se presentan en ambas estructuras con el mismo nombre y el mismo formato.

Ejemplo:

MOVE BY NAME VIEW-ARTICULOS TO ARTICULOS

Los valores de la estructura VIEW-ARTICULOS, se mueven a la de ARTICULOS por nombre.

MOVE EDITED

Se utiliza por lo regular en cambios de formato, siempre se auxilia de máscaras de edición (EM=Edit Mask).

Ejemplo:

```
.....
01 ÑP-FECHA-DATE      (D)
01 ÑP-FECHA-ACTUAL    (A08)
.....
.....
MOVE EDITED '19970411' TO ÑP-FECHA-DATE (EM=YYYYMMDD)
        /* Se convierte una fecha alfanumérica a fecha de tipo date
MOVE EDITED   ÑP-FECHA-DATE (EM=DD/MM/YY) TO
              ÑP-FECHA-ACTUAL
        /* Se convierte una fecha de tipo date a una fecha de tipo
           alfanumérica
        /* La variable ÑP-FECHA-ACTUAL  tendrá como valor lo
           siguiente :  '11/04/97'
```

MOVE RIGHT JUSTIFIED { LEFT }

Esta opción es usada para que el dato sea movido ya sea justificado a la derecha o a la izquierda según sea el caso. Ejemplo:

```
..............
01 ÑP-NOMBRE                    (A30)
..............
..............
MOVE RIGHT JUSTIFIED 'SINERSYS 1997' TO ÑP-NOMBRE
/* Mueve el texto a la derecha del campo
..........................
/* La variable ÑP-NOMBRE tendrá el texto a la derecha : '
   SINERSYS 1997'
```

MOVE ALL

Esta opción mueve repetidamente el valor de un operando1 a un operando2 hasta llenar el operando2 con el operando1.

Ejemplos:

```
..............
01 ÑP-CADENA-ALFA   (A05)
..............
MOVE ALL 'a' TO ÑP-CADENA-ALFA

/* ÑP-CADENA-ALFA   tendrá la cadena siguiente:   'aaaaa'
```

```
0010 ******************
0020 DEFINE DATA
0030 ******************
0040 LOCAL
0050 01 ÑA                  (N3)
0060 01 ÑB                  (A5)
0070 01 ÑC                  (A2)
0080 01 ÑD                  (A7)
0090 01 ÑE                  (N1.0)
0100 01 ÑF                  (A5)
0110 01 ÑG                  (N3.2)
0120 01 ÑH                  (A6)
0130 01 ÑWN-CONT            (N6)
0140 END-DEFINE
0150 ************************************
0160 **        INICIO DE PROGRAMA
0170 ************************************
0180 MOVE 5 TO ÑA
0190 MOVE 'ABCDE' TO ÑB ÑC ÑD
0200 MOVE 1 TO ÑE
0210 MOVE ROUNDED 1.995 TO ÑE
0220 MOVE RIGHT JUSTIFIED 'ABC' TO ÑF
0230 MOVE EDITED '003.45' TO ÑW EM(-999.99)
0240 MOVE EDITED 123.45 (EM=999.99) TO ÑG
0250 ************************************
0260 **        FIN DE PROGRAMA
0270 ************************************
0280 END
```

```
0010 ******************
0020 DEFINE DATA
0030 ******************
0040 LOCAL
0050 01 ÑMA-FECHA                    (A08)
0060 01 ÑWD-FECHA                    (D)
0070 01 ÑMN-A                        (N03)
0080 END-DEFINE
0090 ************************************
0100 **        INICIO DE PROGRAMA
0110 ************************************
0120 IF ÑMA-FECHA EQ MASK (MMDDYYYY)
0130         MOVE EDITED ÑMA-FECHA TO ÑWD-FECHA (EM=DDMMYYYY)
0140 ELSE
0150         REINPUT 'FORMATO DE FECHA INVALIDO, CORRIJA POR FAVOR'
0160         MARK *ÑMA-FECHA ALARM
0170 END-IF
0180 IF ÑMN-A EN MASK (NNN)
0190         REINPUT 'EL VALOR DEBE SER NUMERICO, CORRIJA POR FAVOR'
0200         MARK *ÑMN-A ALARM
0210 END-IF
0220 ************************************
0230 **          FIN DE PROGRAMA
0240 ************************************
0250 END
```

7.4 Uso de las instrucciones EXAMINE, COMPRESS Y SEPARATE.

EXAMINE se utiliza para buscar el contenido de un campo alfanumérico o un campo en forma de arreglo, para reemplazar y/o contar el número de ocurrencias de una cadena de caracteres.

La sentencia COMPRESS es utilizada para transferir o combinar el contenido de dos o más operandos dentro de un solo campo.

SEPARATE es utilizada para separar o transferir el contenido de un operando en dos o más operandos.

Ejemplo:

```
0010 ********************
0020 DEFINE DATA
0030 ********************
0040 LOCAL
0050 01 ARTICULOS VIEW OF ARTICULOS-COMESTIBLES
0060         02 CVE-ART
0070         02 NOMBRE-ART
0080         02 CANT
0090 01 #p-CVE-NOMBRE-ARTICULO                (A20)
0100 01 #p-ARTICULO                           (A07)
0110 01 #p-NOM-ARTICULO                       (A10)
0120 END-DEFINE
0130 ******************************************
0140 **           INICIO DE PROGRAMA
0150 ******************************************
0160 READ (1) ARTICULOS BY CVE-ART
0170      COMPRESS CVE-ART '-' NOMBRE-ART INTO #p-CVE-NOMBRE-
          ARTICULO
0180 END-READ
0190 /*
0200 EXAMINE #p-CVE-NOMBRE-ARTICULO 'LEC1' REPLACE WITH 'LEC3'
0210 EXAMINE #p-CVE-NOMBRE-ARTICULO '-' REPLACE WITH '/'
0220 SEPARATE #p-CVE-NOMBRE-ARTICULO INTO #p-ARTICULO #p-NOM-
     ARTICULO WITH DELIMITER '/'
0230 WRITE '='#p-ARTICULO  '='#p-NOM-ARTICULO
0240 /*  Despliega el articulo y su nombre respectivamente.
0250 ******************************************
0260 **            FIN DE PROGRAMA
0270 ******************************************
0280 END
```

7.5 Comandos de Natural.

A continuación se mencionan algunos de los comandos de NATURAL. Estos comandos se editan desde el prompt de la línea de comandos del editor de programas.

```
>                                        > + PROGRAM.....:
      PROART01        Lib: VENTAS.
```

Comando	Descripción
CATALL	Este comando es utilizado para catalogar todos los objetos de una librería determinada.
CHECK	Se utiliza para revisar la sintaxis del programa.
CLEAR	El comando CLEAR limpia el área de trabajo.
EDIT	Comando utilizado para invocar un objeto natural.
EXECUTE	Utilizado para ejecutar un programa, el comando EXECUTE es opcional , ya que, es suficiente con especificar el nombre del programa en la línea de comando.
FIN	Se utiliza para terminar una sesión de Natural.
GLOBALS	Este comando se utiliza para determinar los parámetros de la sesión de Natural.
HELP	Invoca la ayuda interactiva.
LIST	Lista el área de trabajo fuente u objetos Natural, con este comando no es permisible la actualización en el área de trabajo.
RENUMBER	Renumera el programa que se encuentra actualmente en el área de trabajo.
RUN	Utilizado para compilar y ejecutar un programa fuente.
SAVE	Se utiliza para almacenar el programa fuente.
SCAN	Busca una cadena de caracteres dentro de un objeto de Natural, se puede reemplazar la cadena.
SCRATCH	Borra el código fuente y el código objeto de un objeto NATURAL.
STOW	Este comando es utilizado para almacenar un programa fuente y después de compilarlo salva el código objeto .
XREF	Este comando controla el uso de las referencias cruzadas. Las referencias cruzadas generan automáticamente documentación de los objetos que hacen referencia a un programa. Estos objetos incluyen: campos de una base de datos, subrutinas y mapas.
STRUCT	Ejecuta la indentación de las instrucciones de NATURAL automáticamente e identifica las inconsistencias de estructura.

7.5 Redefinición de variables.

Las variables de usuario, los campos de las vistas o las variables de grupo se pueden redefinir en el Area de Definición de Datos mediante la sentencia REDEFINE. Esta redefinición de variables es muy utilizada en los casos en que se desea tomar una parte de ellas, también cuando la variable esté formando un conjunto de campos, o cuando se quiere tomar como otro tipo de datos (principalmente entre alfanuméricos y numéricos).

Al utilizar la cláusula REDEFINE, se deberán tomar en cuenta los siguientes aspectos:

- La longitud de los campos redefinidos se toma de arriba-abajo y forma la longitud del campo original de izquierda a derecha.
- La longitud total de los campos resultantes de la redefinición debe ser menor o igual a la longitud del campo original.

Sintaxis:

```
Nivel (N)     nombre_campo          (Formato-Longitud)
Nivel (N)     REDEFINE              nombre_campo
              Nivel (N+1)           nombre-campo-redefinido
              (Formato-longitud-redefinido)
              ......                ........
              ..........
              Nivel (N+1)           nombre-campo-redefinido
              (Formato-longitud-redefinido)
```

El nivel (N) se refiere a un dígito de una o dos posiciones usado en conjunción con el nombre del campo (normalmente a nivel 01). Los campos redefinidos deben tener como nivel (N+1), un número consecutivo al del nivel del campo redefinido (normalmente a nivel 02).

Ejemplo:

```
0010 /*
0020 DEFINE DATA LOCAL
0030 01 ÑWA-CVE-ART          (N07)
0040 01 REDEFINE             ÑWA-CVE-ART
0050    02 ÑWA-COD-A         (N02)
0060    02 ÑWA-COD-B         (N05)
0070 01 ÑWA-DESCRIP          (A30)
0080 01 ÑWA-CONTADOR         (N05)
0090 01 ÑWA-DESC-ART         (A30)
0100 01 REDEFINE             ÑWA-DESC-ART
0110    02 ÑWA-DESC-ARRAY    (A01/30)
..... ..............         ........
00150 END-DEFINE
```

V. EJEMPLOS.

Con la finalidad de ejemplificar el uso de las diferentes instrucciones del lenguaje Natural, a continuación se proporcionarán una serie de ejemplos en los cuales se utilizan sentencias de control, diferentes formas de lectura a archivos de la base de datos, lectura y escritura a archivos secuenciales, asignación de atributos a los campos, así como el uso de máscaras de edición.

Ejemplo 1.

El siguiente programa tiene como objetivo el ejemplificar el uso de atributos, para lo cual se definen dos variables de control (tipo 'C'), previamente asignadas a los campos MA-CVE-ART y MA-PREC-VENTA dentro del mapa. Durante la ejecución del programa, mediante el uso de estas dos variables de control (WA-ATTR y WB-ATTR), los campos MA-CVE-ART y MA-PREC-VENTA cambian su estado de 'Temporalmente Protegido' (AD=P, únicamente para desplegar contenido del mismo), a 'Desprotegido' (AD=I, modificable).

La función del programa es cancelar artículos en base a su precio de venta.

```
0010 ************************************************************
0020 **  SISTEMA :       INVENTARIO DE ARTICULOS
0030 **  PROGRAMA :      INVART06
0040 **  DESCRIPCION :   CANCELACION DE ARTICULOS EN BASE AL
0050 **                  PRECIO DE VENTA
0060 ************************************************************
0070 DEFINE DATA
0080 LOCAL
0090 * ---------------- VARIABLES AUXILIARES ------------------- *
0100 01 ART VIEW OF ARTICULOS-COMESTIBLES
0110    02 CVE-ART
0120    02 NOMBRE-ART
0130    02 CVE-PRESNT
0140    02 CANT
0150    02 PRECIO-VENTA  (1:10)
0160    03 PRECIO-PUBLICO
0170    03 PRECIO-DISTRIBUIDOR
0180 * ------------------- VARIABLES DEL MAPA ------------------ *
0190 01 MAPA-ARTICULOS
0200    02 MA-PROGRAM          (A08)
0210    02 MA-CVE-ART          (A07)
0220    02 MA-NOMBRE-ART               (A10)
0230    02 MA-CVE-PRESNT               (A05)
0240    02 MA-DESCRIP          (A30)
0250    02 MA-CANT             (N05)
0260    02 MA-PREC-VEN         (N16/1:10)
0270    02 REDEFINE            MA-PREC-VEN
0280       03 OCCUR-PREC              (10)
0290          04 MA-PREC-PUB        (N6.2)
0300          04 MA-PREC-DIST       (N6.2)
0310    * ------------------ VARIABLES AUXILIARES ----------------*
0320 01 ÑWA-FECHA-ACTUAL                (A08)
0330 01 ÑWA-BANDERA1           (L)
0340 01 ÑWA-BANDERA2           (L)
0345 01 ÑWA-BANDERA3           (L)
0350 01 ÑWA-ART-PRESNT               (A12)
0360 01 REDEFINE                      ÑWA-ART-PRESNT
0370    02 ÑWA-ART              (A07)
0380    02 ÑWA-PRESNT           (A05)
0390 01 WA-ATTR                      (C)
0400 01 WB-ATTR                      (C)
0410 END-DEFINE
0420 * ------------------------ PROCESO -------------------------*
0430 RESET MAPA-ARTICULOS
0440 MOVE *PROGRAM TO MA-PROGRAM
0450 MOVE *DATN TO ÑWA-FECHA-ACTUAL
0460 REPEAT
0470          PF1 NAMED 'AYUDA'
0480          PF2 NAMED '-----'
0490          PF3 NAMED 'MENU'
0500          PF4 NAMED '-----'
0510          PF5 NAMED 'MDF.'
0520          PF6 NAMED 'SIG.'
0530          PF7 NAMED '-----'
0540          PF8 NAMED '-----'
0550          PF9 NAMED '-----'
0560          PF10 NAMED '-----'
0570          PF11 NAMED '-----'
0580          PF12 NAMED 'FIN'
0590          PF24 NAMED 'FIN'
0600          SET CONTROL 'WBL80C24M'
0610          INPUT USING MAP 'ARTIC010'
0620          DECIDE ON FIRST VALUE OF *PF-KEY
0630          VALUE  'ENTR'
0640                 IF ÑWN-BANDERA1 EQ TRUE THEN
0650                        IF ÑWN-BANDERA2 = FALSE THEN
0660                               MOVE (AD=I) TO WA-ATTR
0670                               MOVE (AD=P) TO WB-ATTR
0680                               RESET MAPA-ARTICULOS
0690                        ELSE
0700                               PERFORM ACTUALIZAR-ARTICULO
0710                        END-IF
0720                 ELSE
0730                        PERFORM CONSULTAR-ARTICULO
0740                 END-IF
0750          VALUE   'PF1'
0760                 PERFORM AYUDA-GENERAL
0770          VALUE  'PF5'
0780                 IF ÑWN-BANDERA1 EQ TRUE THEN
0790                        PERFORM ACTUALIZAR-ARTICULO
0800                 ELSE
0810                        IF MA-CVE-ART NE '   ' THEN
0820                               REINPUT 'DIGITE <ENTER>'
0830                               MARK *MA-CVE-ART'
0840                        END-IF
```


000063

```
0850                END-IF
0860          VALUE 'PF6'
0870                MOVE (AD=I) TO WA-ATTR
0880                MOVE (AD=P) TO WB-ATTR
0890                RESET MAPA-ARTICULOS
0900          VALUE 'PF12'  'PF24'
0910                TERMINATE
0920          NONE
0930                REINPUT 'TECLA DE FUNCION INVALIDA'
0940                MARK 1
0950   END-REPEAT
0960 *
0970 * ------------------------------------
0980 DEFINE SUBROUTINE CONSULTAR-ARTICULO
0990 * ------------------------------------
1000 RESET #WN-BANDERA1
1010 MOVE (AD=P) TO WA-ATTR
1020 MOVE (AD=I) TO WB-ATTR
1030 IF MA-CVE-ART = '   ' THEN
1040          REINPUT 'CLAVE DE ARTICULO REQUERIDA'
1050          MARK *MA-CVE-ART
1060 END-IF
1070 RESET #WN-BANDERA3
1080 FOR I = 1 TO 10
1090          IF MA-PREC-VEN(I) NE ' ' THEN
1100                MOVE TRUE TO #WN-BANDERA3
1110          END-IF
1120 END-FOR
1130 IF #WN-BANDERA3 EQ FALSE AND MA-CVE-ART NE ' ' THEN
1140          REINPUT 'PROPORCIONAR PRECIO DE VENTA'
1150          MARK *MA-PREC-PUB(1)
1160   END-IF
1170 MOVE RIGHT JUSTIFIED MA-CVE-ART TO MA-CVE-ART
1180   FIND ART WITH CVE-ART EQ MA-CVE-ART
1190          IF NO RECORDS FOUND
1200             REINPUT 'NO EXISTE REGISTRO PARA ESTA CVE.'
1210             MARK *MA-CVE-ART
1220          END-NOREC
1230          MOVE CVE-ART  TO MA-CVE-ART
1240          MOVE NOMBRE-ART TO MA-NOMBRE-ART
1250          MOVE CVE-PRESNT TO MA-CVE-PRESNT
1260          MOVE DESCRIP TO MA-DESCRIP
1270          MOVE CANT TO MA-CANT
1280          FOR I = 1 TO 10
1290                MOVE PRECIO-VENTA(I) TO MA-PREC-VEN
1300          END-FOR
1310   END-FIND
1320 MOVE TRUE TO #WN-BANDERA1
1330 END-SUBROUTINE
1340 *
1350 * ------------------------------------------
1360 DEFINE ACTUALIZAR-ARTICULO
1370 * ------------------------------------------
1380 MOVE (AD=I) TO WA-ATTR
1390 MOVE (AD=P) TO WB-ATTR
1400 FIND ART WITH CVE-ART EQ MA-CVE-ART
1410          IF NO RECORDS FOUND
1420             REINPUT 'NO EXISTE REGISTRO PARA ESTA CVE.'
1430          MARK *MA-CVE-ART
1440          END-NOREC
1450          FOR I = 1 TO 10
1460                MOVE MA-PREC-VEN(I) TO PRECIO-VENTA(I)
1470          END-FOR
1480          UPDATE(1400)
1490          END TRANSACTION
1500 END-FIND
1510 PERFORM DESPLEGAR-CANCELACION
1520 RESET MAPA #WN-BANDERA1 #WN-BANDERA2
1530          #WN-BANDERA3
1540 END-SUBROUTINE
1550 *
1560 * ------------------------------------------
1570 DEFINE SUBROUTINE DESPLEGAR-CANCELACION
1580 * ------------------------------------------
1590 SET KEY OFF
1600 SET KEY ENTR
1610 SET CONTROL 'WL40C1000/25F'
1620 INPUT '                                        '
1630   /   '       ARTICULO ** CANCELADO **         '
1640   /   '                                        '
1650   /   '   PRESIONE <ENTER> PARA CONTINUAR      '
1660   /   '                                        '
1670 *
1680 * -------------------------------
1690 DEFINE SUBROUTINE AYUDA-GENERAL
1700 * -------------------------------
1710 SET KEY OFF
1720 SET CONTROL 'WL60C1003/19F'
1730 INPUT '          CANCELACION DE ARTICULOS      '
1740   /   '                                        '
1750   /   '  PROPORCIONE EL NUMERO DE ARTICULO,    '
1760   /   '     <PF1> ---> AYUDA                   '
1770   /   '     <PF3> ---> TERMINAR PROGRAMA       '
1780   /   '     <PF5> ---> CONFIRMAR CANCELACION   '
1790   /   '                                        '
1800   /   '    PRESIONE <ENTER> PARA CONTINUAR     '
1810 SET KEY ON
1820 END-SUBROUTINE
1830 *
1840 END
```